DAG Graph for IoT performance evaluation: Task Partitioning and Co-design Methodology

1st Radia Bendimerad
E.Polytechnique, Lab SERCOM, Marsa
Tunis, Tunisie
radia.bendimerad@fst.utm.tn

2nd Kamel Smiri
E.Polytechnique, Lab SERCOM, Marsa
Tunis, Tunisie
smiri_kamel@yahoo.fr

3rd AbderrazekJemai
E.Polytechnique, Labo SERCOM, Marsa
Tunis, Tunisie
abderrazekjemai@yahoo.co.uk

***Abstract*—IoT systems are getting more complex with the exponential growth of embedded applications which reduces their robustness and compromises the quality of service. Some solutions based on performance estimation have been proposed to overcome this problem. However they are manually processed and limited such as complex heuristics and random process. One solution is to implement an automatic scheduling approach that meets the constraints of application and making compromise between the needs imposed by it and the offers proposed by the target IoT in order to obtain gains in term of total equitable distribution of workload over the various resources available in the system, reduce processing time and achieve performance satisfaction for a multiprocessing IoT system. In this paper, we propose an original methodology which deals with both Task Partitioning and CO-design Methodology that we called (TPCOM) that allows a good scheduling from a SDF (Synchronous Data Flow) graph of the application. An experiment study on the MJPEG reference application is conducted in order to illustrate the effectiveness of our approach.**

***Index Terms*—Internet of things (IoT), Codesign, Synchronous data flow (SDF), Directed acyclic graph (DAG), Performance.**

I. INTRODUCTION

Nowadays, the Internet of Things has became more and more useful, it improves several characteristics by defining variant applications in different fields. Also, it encourages the intensive use of heterogenous devices containing multiprocessors system on chip (MPSoCs) that enable the integration of multiple microprocessors and allow a good treatment capacity, they have been recognized as the most powerful and interesting embedded systems technology for the development of the Internet of Things as described in [1] [2]. This provides exponential growth of acquired/transmitted data raises bandwidth and latency issues as well as sufficient time data monitoring. For applications requiring real-time data processing, such as video surveillance, it encounters the following problems:

- The multiple constraints and QoS requirements of users (application execution time).
- The heterogeneity of the components available to the system (components with different resources and linked by heterogeneous technologies of different capacities).
- The transfer of large volumes of data (resources may be geographically distant from each other, which can cause problems for devices that have a large amount of data for limited storage space).

The solution expected in the perspective to ensure the proper running of applications and therefore the system performance is based on task partitioning methods as in [3]. and can be summarized in the optimization of the execution time. For this end, many previous research has focused on the hardware and software co-design, where the two parts design are jointly linked as well as the partitioning and scheduling application tasks as notified in [4][5]. In [6], they propose an optimization of the hardware/software partitioning of very low-power wireless receiver digital signal processing operators (parts of the DSP chain of the transceiver), defined by software, designed for IoT applications using an open-source 32-bit RISC-V. Reference [7] aimed to enable their heterogeneous IoT system, consisting on processor and reconfigurable hardware, transmitter-receiver, to achieve high data rates with low power consumption. The approach consists in creating Simulink model variants for the transmitter and receiver: Rx (receiver) which each implement a different number of functional blocks in HW and SW where data is moved manually from the Software part to the Hardware part. For each processing chain receiver (Rx) and Transceiver (Tx), the HW-SW Co-design is explored by creating a number of variants Tansceiver. Also, as defined in [8] the use of different manual configurations to finally select the best one in order to optimize a flexible hardware/software co-design for the implementation of a gateway at the Edge of the network, for simultaneous compression and encryption with DCT method plus spectral filtering of images on a Zynq platform in the context of IoMT (Internet of Multimedia Things).
Most of the existing work in IoT systems that we briefly review here provides solutions for manual HW/SW partitioning based on the system. Also, the problem of partitioning and scheduling tasks in an application has been the subject of several studies on multiprocessor systems but has not been studied in detail other systems such as IoT (Internet of things). So that, the proposed contributions of this paper are listed as follows:

1) Modeling the application and target platform via SDF graphs as explained in[10]

2) Transform the SDF graph into DAG (directed acyclic graph).

3) Propose labeling and Classification of tasks into two groups for pure SW scheduling and SW/HW scheduling.

4) Identify the priority between tasks in the graph.

5) Distribute tasks over available resources under several execution constraints.

6) Check the constraints at each partitioning and update the storage space allocations.

The Table I prensents a synthesis of previous works belonged to our field. We take some references that we introduced in the state of art. The synthesis is based on the following elements:

- target architecture: IoT or MPSoC.
- data volume.

In this work, we introduce a new task partitioning approach called TPCOM (task partioning and co-design mothodology) for scheduling in SDF graph-based IoTs using the modeling, co-design and partitioning techniques in order to have an automatic configuration and processing of data in real time. We experiment this approach to test, validate results and demonstrate the effectiveness and robustness of our strategy in term of execution time and better system performance. The remainder of this paper is organized as follows: Section II surveys problematic formalization and nominates solution. Section III discusses the contribution overview and proposed strategy. Section IV considers a case study with obtained gains. Section V concludes with a discussion of future works.

II. Formalization of approach

In this section, we describe what is the Internet of things by modeling formally all its constituents : the hardware architecture supporting the execution with its different components and the application with its constraints, in addition to introducing some concepts such as application techniques modelling in order to better understand our methodology. The Internet of things is, as its name implies, a set of heterogeneous objects communicating through the Internet via guidelines, protocols and standards that simplify the implementation of IoT applications as shown in [11]. Fig. 1 illustrates an example of the IoT system with different components.

An IoT system is a combination of a hardware platform and a software application.

A. Hardware architecture

Our IoT system includes a set of electrical components that we call board with $B = \{B1, \ldots, Bm\}$ such as SoC (Sytem On Chip) connected to each other via communication networks. A communication C is annotated by Cn which designs the bandwidth and whose rate differs between each pair of Board. We assume that devices are heterogenous i.e the components are grouped into categories. Each category has its own characteristics such as execution time, processor behaviour : in fact, some processors of our components system are composed of software behaviour only and others composed of software and hardware behaviour which means that there is Hardware tasks and they are generally provided as IPs (Intellectual Property) written in a hardware description language (e.g., VHDL, verilog) and implemented by FPGAs. The proposed methodology remains valid with homogenous components too.

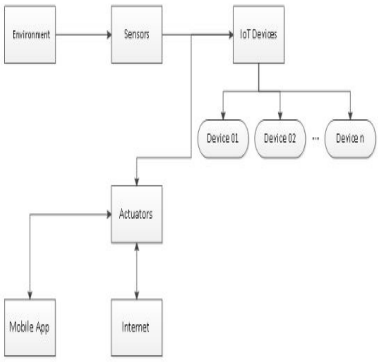

Fig. 1. Transformation of SDF graph into a DAG

B. Application

The application is a set of tasks to be executed and modeled by Synchronous Data Flow (SDF) graphs used to design simultaneous multimedia applications implemented on multiprocessor-on-chip systems as noticed in [12]. A common way to schedule SDF graphics tasks on multiple processors is to first convert the SDF graph into a directed acyclic precedence graph called DAG as in [13]

1) Synchronous Data Flow (SDF): SDF is a graph with actors as vertices and channels as arcs. Actors represent the basic parts of an application that must be executed. Channels represent the data dependencies between actors. The execution of an actor is designed by an actor who fires tokens[4]. The SDF can be classified into main categories, among which we mention the one that we are using in this work : HSDF (homogeneous SDF) is an SDF where each actor produces and consumes exactly one token as indicated in [14]. $G_S(V, E, E, p, c)$ represent the SDF graph where each of these parameters is explained below:

- V is the set of tasks noted by the nodes in the graph.
- E is the set of arcs that connect the nodes. They determine the dependencies between tasks.
- $p(e)$ is the number of data tokens produced by the source node of arc e.
- $c(e)$ is the number of data tokens consumed by the destination node of arc e.

2) Directed Acyclic Graph (DAG): An oriented graph is a graph in which each pair of nodes is connected by a directed arc. each actor consumes and produces a single data token. It can be represented by a notation $G_D(V, E)$ where :

TABLE I
SYNTHESIS TABLE

	data volume	Processors number	FPGA	Processors size	Co-design performance	Communication
[6]	IoT	Not mentioned	Not mentioned	Yes	Lack of model	Not mentioned
[7]	IoT	Not mentioned	Not mentioned	Yes	Lack of model	Not mentioned
[8]	IoT	No	Yes	No	Lack of model	Yes
[9]	IoT	No	Not mentioned	No	Lack of model	Yes
[18]	IoT	Yes	Yes	Yes	Lack of model	No

- V is a set of nodes representing the processes
- E is a set of directed arcs connecting the nodes. They describe a dependency relationship, i.e. a node connected to its predecessor cannot run unless the parent process is completed.

In the aim of transforming the SDF into a DAG graph we need :

- The topological matrix: which stores the information of each arc in the synchronous data flow graph : is the size matrix, where each row corresponds to an arc (e) in the SDF graph and each column corresponds to a node (v).The values on each arc indicate the tokens produced and consumed by each vertex or node i on arc j as in [15].
- The BVR (Basic Vector Repetition) repetition vector: number of copies for an actor (node) to be scheduled for each iteration. as in [16].

C. Objective function

is to optimize the overall execution time generated by the allocation of available resources so that the value is minimal while respecting functional and non-functional constraints and taking into account:

- Software SW and Hardware HW execution time ET of existant tasks, as in Equation (1).

$$ET = H_{ET} + S_{ET} \tag{1}$$

  where H_{ET} refers to HW execution time and S_{ET} refers to SW execution time.
- Transfer time due to data exchanges between the different electronic boards used during the assignment of DAG tasks or data exchanges between the SW and HW parts. The transfer time $TT(B(t_i), B(t_j))$ is given by the following equation reported in [17], as defined in Equation (2).

$$TT(B(t_i), B(t_j)) = \frac{data[i,j]}{C[B(t_i), B(t_j)]} \tag{2}$$

  where $data[i,j]$ refers to the amount of data to be transfered from i to j and C is the bandwidth allowed between two electronic boards $B(t_i)$ and $B(t_j)$ that execute task t_i and task t_j.

As a result, the objective function is the sum of task execution time and all transfers performed on the tasks executed as in Equation (3).

$$F(ET) = \begin{cases} min\{ET(t_i)\} & \textit{if} \quad TT = 0 \\ \textit{else} & \\ min\{ET(t_i) + TT(B(t_i), B(t_j))\} & \end{cases} \tag{3}$$

The goal of the current paper is to provide a solution that places the tasks in the related devices under functionnal and non functionnal constraints. Those must be considered by the objective function :

- Functional constraints : whose source constraints are directly related to the operation and behavior of the application.
- Non functional constraints : where the sources of constraints are related to standardizations and hardware architectures.

Problem : face to the platform of the IoT system considered with its different boards, their software or software/harware behaviour, arises the problem of resource allocation (determining the number of resources to be used to perform each task) and task scheduling (determining the order in which tasks are performed on each resource) of an application represented with a SDF graph that is a NP-difficult problem. Therefore, most of the proposed solutions are heuristics that produce good results. However, they are more expensive in terms of execution time and algorithmic complexity since they are not linear problem's resolution.

III. CONTRIBUTION OVERVIEW

In this section, we present an overview about the proposed methodology based on two concepts : Task Partitioning and CO-design Methodology (TPCOM). We first present the first part of the methodolody then we show how the co-design TPCOM improves the performance.

A. Task Partitioning:

It consists on three main phases.

1) Phase 1 (Application and hardware platform modelling) : firstly, we represent of the IoT system and application according to the SDF model [10].
2) Phase 2 (Transformation of SDF graph into a DAG) : Directed Acyclic Graph DAG is very useful for scheduling tasks in iterations without violating dependencies between them because in a SDF, each actor is able to produce and consume several data tokens, which

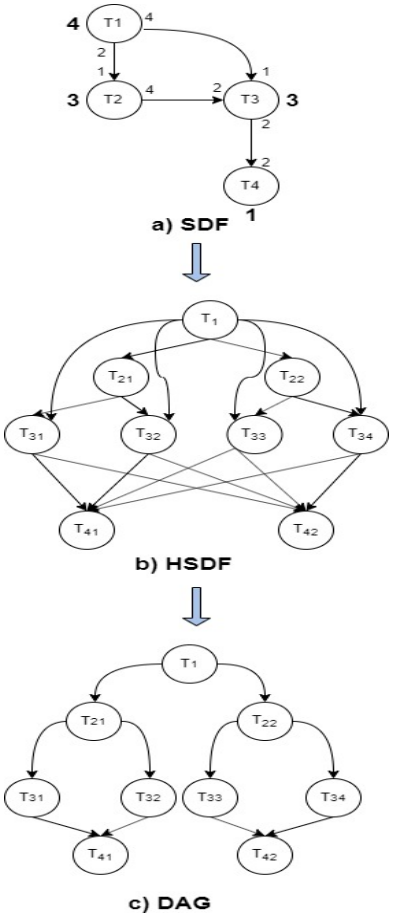

Fig. 2. Transformation of SDF graph into a DAG

makes this operation difficult. For transition from SDF to DAG.The first step of transformation of the SDF graph [15] is a conversion of the SDF into an HSDF, for this purpose we associate to the SDF graph the topology matrix from which we construct the HSDF graph (positive value corresponds to the number of child nodes, the negative value corresponds to the arcs from the parent nodes. The second step is the generation of the DAG by removing redundant arcs as shown in [16]. The last step is to ignore FIFOs with the initial tokens in the SDF single rate graph.

3) Phase 3 (Ressource selection) : tasks at each level are assigned to the available resources according to the estimated time required to perform each task on each available component.

The test performed on this method has given satisfying results regarding the optimization of the execution time. However, we noticed a lack of exploitation of the resources. Indeed, hardware accelerators that can further optimize execution time were not used. This is why we expect better performance when introducing co-design.

B. Task partitioning and Co-design Methodology TPCOM:

In this part, we keep the phases 1 and 2 of Task Partitionning method, add three more phases after phase 2 and modify the last phase which consists on resource selection. The goal of our methodology is to achieve a good distribution of the application tasks on the different resources available and within these resources in order to obtain automatic scheduling and optimal execution time, so that all the system requirements and the design constraints are respected. To do this, we combine the use of task parallelism with hardware execution to further accelerate the process by applying loop-level partitioning. The idea is to classify tasks into several groups. Tasks with loops of high execution time require a hardware accelerator and must run on the component with an FPGA part. Tasks whose loops do not consume much time for execution do not require a hardware accelerator and will therefore be performed in the order of priority. All the phases of our methodology are depicted in Figure 4.

In detail, the phases 3, 4, 5 and 6 are :

- Phase 3 (Controller unit): The SDF transformed into a DAG will pass through the controller unit, the latter will allow the classification of the different tasks into two groups: the tasks requiring a hardware accelerator and the second group which includes the rest of tasks. The controller operation consists mainly on:
  - Loop partitioning : divide the graph into several distinct levels. The principle is to go through the graph from top to bottom in order to have different levels so that the tasks belonging to each level are independent of each other but remain identical for each level. Example for the graph in figure 3.b the different levels are l_i containing tasks t_i are:
    $l_1 = \{t_1\}$
    $l_2 = \{t_{21}; t_{22}\}$
    $l_3 = \{t_{31}; t_{32}; t_{33}; t_{34}\}$
    $l_4 = \{t_{41}; t_{42}\}$
  - Tasks classification : Split tasks into subtasks in order to extract loops and profile candidate [20] from a single node per level (nodes of the same level are identical in the case of transformed SDF into DAG) to identify and select those that increase the application's execution time. With this controller, we will increase the occupancy time of IoT components via hardware accelerators in order to deploy resources more optimally. As a result we obtain two groups:
    * Group 01: tasks performed on components incorporating hardware accelerators.
    * Group 02: the rest of the tasks.

    At the time of allocation, the priority of the component selection is given to group 01 containing the tasks requiring a hardware accelerator.
- Phase 4 (Labeled DAG) : Return to the original DAG and label tasks according to their membership in one of the two groups.

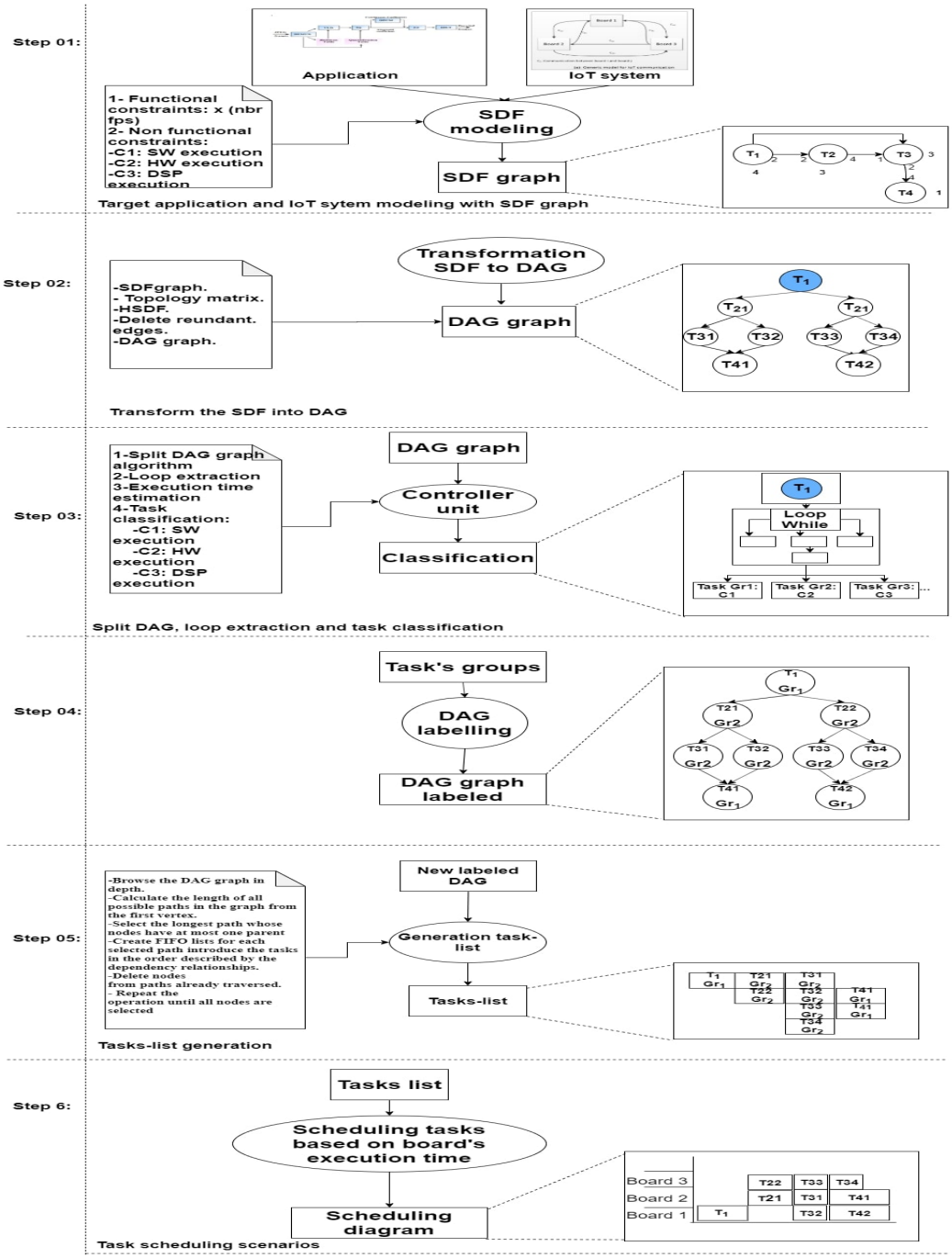

Fig. 3. TPCOM flow diagram

- Phase 5 (List scheduling and Task prioritization): In this step, we again explore our labelled DAG and create lists where we order the tasks of the labelled DAG. To do this, we proposed a strategy consisting of the following steps:
  - Browse the DAG graph in depth.
  - Calculate the length of all possible paths in the graph from the first vertex.
  - Select the longest path whose nodes have at most one parent: if a node is connected to two or more parent nodes, this will cause confusion because it must be checked that all predecessor nodes have been selected as well.
  - Create FIFO lists for each selected path and introduce the tasks in the order described by the dependency relationships.
  - Delete nodes from paths already traversed.
  - Repeat the operation until all nodes are selected.

After exploring the entire graph, we obtain several lists,

each containing at least one labelled task.

- Phase 6 (Selection of adequate resources for tasks) : The choice of resources is not random but consists in choosing from the FIFO lists the one whose first task has the TEM (minimum execution time) via an estimated matrix of the execution time of the tasks on each available resource by considering the labelling of the tasks, also via the flow matrix in case of data transfer from one resource to another.

The following algorithm 1 implements the described approach by considering the objective function:

Algorithm 1: TPCOM algorithm

```
Model the SDF graph;
Transform the SDF into DAG;
Split the graph into levels;
for Level=1 To l do
    consider one node only;
    Split the task's node into sub-tasks;
    foreach SubTask do
        Extract loops;
        calculate the execution time;
        if Time > x then
            Node assignment to Gr_01;
        else
            Node assignment to Gr_01;
        end
        Label the DAG (Gr_01 or Gr_02);
        Transform the DAG in FIFO list for scheduling;
    end
end
foreach Element ∈ List do
    Extract the element label;
    if Gr_01 then
        execute Hardware accelerator;
    else
        estimate the minimal execution;
        if Blocked Component then
            Transfert to another component using
             equation (2);
        else
            execute;
        end
    end
end
forall tasks t_i do
    Assign task using equation (3);
end
```

IV. Implementation and Experimentation

A. Case study

The obtained results from the proposed strategy are illustrated by a case study that we prensent. We consider for our IoT hardware architecture three different electrical boards and the MJPEG application as follows:

1) Hardware IoT platform: We consider three heterogenous boards, these boards are these components are of the type : Xilinx, Arduino and Rasbperry Pi, as shown in the figure 4.

2) MJPEG application: The Motion-JPEG or MJPEG decoder is a multimedia application whose components are used in many image and video processing algorithms. It is a video codec that compresses images one by one into JPEG. The first functional block, DEMUX, sends the input flow to the other blocks. VLD performs variable length Huffman decoding. ZZ reorganizes the flow of coefficients. IQ performs the reverse quantification. IDCT performs the inverse discrete cosine transform. LIBU is not a JPEG operation, but it is necessary to adapt the pixel stream to a given RAMDAC device controller output.

In this paper, MJPEG application that is shown in the figure 5 is choosen to evaluate our approach. It consists in finding a solution for optimizing execution time while respecting the constraints related to the application and those related to the hardware architecture.

- application constraint : The constraint of 25 frames/second decoding means that the time of decoding of an image must be less than 40 ms.
- Hardware platform constraint : CPU number and capacity, hardware accelerator presence or not.

3) Validation of TPCOM approach : After the application and system are modeled in $phase01$ we illustrate the approach in next phases.

- Phase 2 (Transform the SDF graph into a DAG) : The second phase of our approach is the transformation of the SDF model into a DAG model. Then it is a matter of split the graph into levels, and extracting the set of loops from the application. Since most nodes for the same level are identical, it is sufficient to extract loops from a single node. We then evaluate the execution time of each loop according to which a controller will classify the tasks in the corresponding group. The figure 6 shows the SDF model transformation of MJPEG into a DAG and loop extraction.

- Phase 3 (loops synthesis): The second column of Table II shows the number of loops from which we profile the execution time. The third column shows the number of loops that contribute more than 1% of the total execution time of the program. The fourth column shows the percentage contribution of these loops to the total execution time. Among these loops we choose those whose execution time is greater than x (with x is the umber of frames per second or $25fps$), then we obtain two groups :
  - Group 01: concerns tasks whose execution time is greater than x.
  - Group 02: the rest of the tasks.

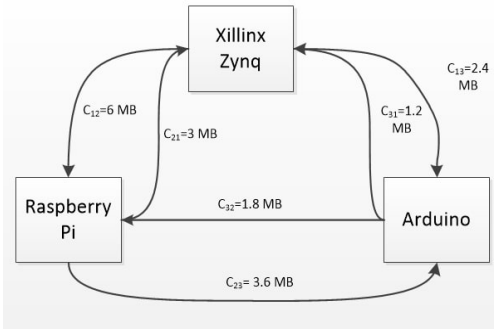

Fig. 4. IoT architecture example

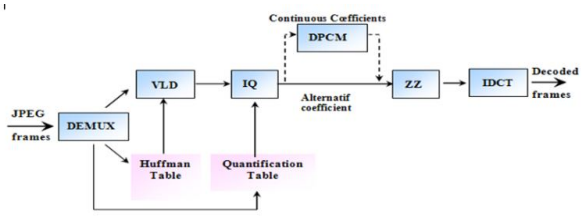

Fig. 5. MJPEG decoder

We found that these loops with high execution time were mainly concentrated in the IDCT task.

- **Phase 4** (Labled DAG) : the old DAG is browsed to make a label for ech task.
- **Phase 5** (Task list priority) : From the classification obtained previously, we start by building the labelled graph in order to extract the task lists necessary for scheduling. The figure 7 shows the labelled DAG and the order of execution list of tasks.
- **Phase 6** (Resource selection): for the selection of resources, Table III is used to estimate the execution time of tasks on each component, so for each task not belonging to Group 01, we choose the component according to the minimum execution time and its availability. Taking into account the estimated execution time and the DAG labelling, the obtained scheduling diagram is presented in the figure 8:

B. Performance Evaluation

In this section our main objective is to evaluate our experiences and their results. Indeed, our approach to the performance analysis of IoT systems based on SDF graphs is interesting because it allows us to take advantage of resources in a more optimal way, taking into account the constraints imposed by the application and the platform. In addition, this approach gives more precise results in the execution time calculated in relation to the simulation results since we have exploited each of the components' functionalities (SW for components that have processors only or SW and HW for IoT components that have the FPGA part) and dynamic scheduling has allowed us to obtain a minimum execution time.

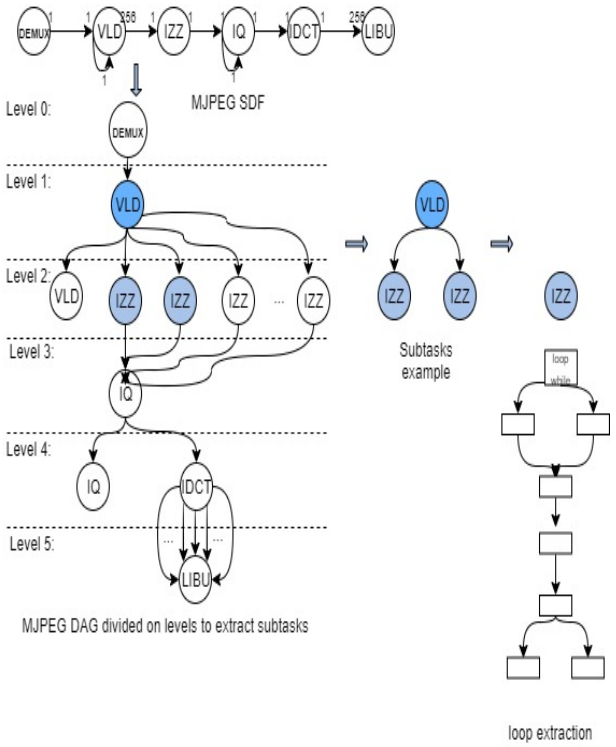

Fig. 6. transformation of SDF into DAG and loop's extraction

TABLE II
LOOP'S PROFILING

	Loop	Loop * %	Total (%) 1%
MJPEG encoder	165	14	65

V. CONCLUSION

In this work, we present a method for performance analysis based on SDF graphs. It is a method for modeling, co-design partitioning and scheduling the tasks of an application within an IoT system called TPCOM methodology. Our new approach thant contains our original contributions have been presented in previous sections. This new approach is based on 6 steps:1) system specification and application modeling with SDF graph, 2) transformation of the SDF graph to the DAG graph, 3) classification tasks onto several groups, 4)labelling the DAG 5) generation on task-list and 6) selection of a scheduling scenario for the complete IoT system (application and hardware architecture). With this technique we offer an optimal use of the IoT components. We can determine at an early stage for a given task the appropriate resource and when it should migrate into the hardware. This technique allows an improvement in the performance of the overall execution time, as well as Software/Hardware partitioning and optimal scheduling of tasks in IoTs, taking into account the constraints described by the application and the target platform. An experiment is done on the MJPEG decoder to illustrate the effectiveness of our technique.

In this paper, we study the problem of task scheduling in

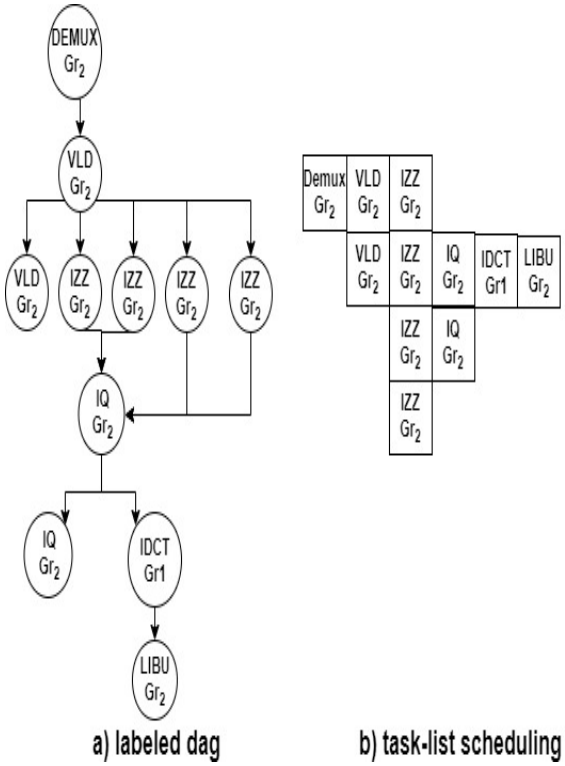

Fig. 7. Labeled DAG and task list scheduling

TABLE III
ESTIMATED TIME TO COMPLETE MJPEG TASKS (TIME IN SEC)

	Zedboard	Raspberry Pi	Arduino
DEMUX Gr_2	1	2	4
VLD Gr2	15	10	16
IZZ Gr_2	9	7	5
IQ Gr_2	2	5	4
IDCT Gr_1	20	23	18
LIBU Gr_2	3	5	6

the context of IoT systems in order to find a better scheduling taking into account the resources and the constraints at hardware and software levels.

However, our approach has some limitations that we will try to improve in future work: - Graphical modeling of the application via DAGs can become complex in the case of massive data. - The degree of complexity of the applications considered for our approach is of medium complexity. - Our solution targets static order scheduling. As perspectives, we plan to focus on problems that components may encounter as failure problem. We also plane to improve our method and extend it to intelligent and more complex IoT systems such as smart buildings, smart grids...etc.

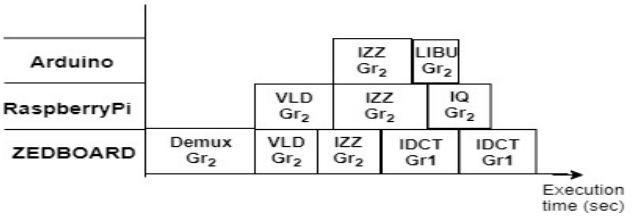

Fig. 8. scheduling diagram

REFERENCES

[1] F. Samie, L. Bauer, and J. Henkel, " IoT technologies for embedded computing: a survey" IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, New York, NY, USA, Article No. 8, October 2016.

[2] P. D. Rosero-MontalvoEmail, V. F. López, B. A. RoseroEdgar, D. JaramilloJorge, A. Caraguay, J. Pijal-RojasD, and H. Peluffo-Ordóñez. "l'Intelligence in Embedded Systems: Overview and Applications" Volume 1. Intelligence in Embedded Systems, 2019.

[3] R. Zaman Khan, J. Ali. "Task Partitioning Strategy with Minimum Time (TPSMT) in Distributed Parallel Computing Environment". International Journal of Application or Innovation in Engineering and Management (IJAIEM), 2013.

[4] M. Ammar, G. Russello, B. Crispoa. "Internet of Things: A survey on the security of IoT frameworks". Journal of Information Security and Applications. Vol. 38, pp. 8-27, 2017.

[5] R. Kramer, R. Gupta, and M.L. Soffa. "The combining DAG: a technique for parallel data flow analysis". IEEE Transactions on Parallel and Distributed Systems (TPDS). Vol. 5 , Issue. 8 , 1994.

[6] H. Belhadj Amor, and C. Bernier. "Software-Hardware Co-Design of Multi-Standard Digital Baseband Processor for IoT". Design, Automation and Test in Europe Conference, and Exhibition. IEEE, Florence, May 2019.

[7] B. Drozdenko, M. Zimmermann, T. Dao, K. Chowdhury, and M. Leeser. "Hardware-Software Codesign of Wireless Transceivers on Zynq Heterogeneous Systems". IEEE. Vol. 6, January 2017.

[8] Maher Jridi, T. C. "SoC-Based Edge Computing Gateway in the Context of the Internet of Multimedia Things: Experimental Platform". Low-Power Electronic Circuits for Monolithic Smart Wireless Sensors. Vol. 8, 2018.

[9] T. Adegbija, A. Rogacs, C. Patel, and G. Ross. "Microprocessor Optimizations for the Internet of Things: A Survey". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. vol. 37, pp 7-20, 2018.

[10] K. SMIRI, and A. Jemai. "NoC-MPSoC Performance Estimation with Synchronous Data Flow (SDF) Graphs". Autonomous and Intelligent Systems. vol. 6752, pp 406-415, 2011.

[11] A. Al-Fuqaha, M. Guizani and M.Mohammadi. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". IEEE. vol. 17, pp. 2347 - 2376, 2015.

[12] A.H. Ghamarian, M. G., Basten, T., Theelen, B., Mousavi, M., and S. Stuijk. "Liveness and Boundedness of Synchronous Data Flow Graphs". IEEE, 2006

[13] R. Tariq, F. Aadil and Muhammad.F.M. "Directed Acyclic Graph Based Task Scheduling Algorithm for Heterogeneous Systems". Proceedings of SAI Intelligent Systems, vol. 869, pp 936-947, 2018.

[14] K. Desnos, M. Pelcat, JF. Nezan, S Aridhi. Memory Analysis and Optimized Allocation of Dataflow Applications on Shared-Memory MPSoCs. Journal of Signal Processing Systems. vol. 80, pp 19–37, 2014.

[15] Pullaguntla, R. Krishna. ROTATION SCHEDULING ON SYNCHRONOUS DATA FLOW GRAPHS, 2008.

[16] D Li, J Wu. Energy-Aware Scheduling for Acyclic Synchronous Data Flows on multiprocessors. Interconnection Networks. vol. 14, 2014.

[17] Kahina Bessai, and F. Charoy. Business Process Tasks-Assignment and Resource Allocation in Crowdsourcing Context. IEEE . 2016.

[18] DG Costa, C Duran-Faundez. Open-Source Electronics Platforms as Enabling Technologies for Smart Cities: Recent Developments and Perspectives. electronics. vol.7, 2018.

[19] I Ghribi, RB Abdallah, M Khalgui. R-Codesign: Codesign MethodologyforReal-Time Reconfigurable Embedded Systems Under Energy Constraints, IEEE, 2018.