Analyzing Throughput for Cyber-Physical Systems modeled with Synchronous Dataflow

Philippe Glanon, Selma Azaiez, Chokri Mraidha

CEA-LIST Saclay, Gif-sur-yvette, France
surname.name@cea.fr

Abstract. Cyber-Physical System (CPS) is a critical system in which timing performance is often required. Throughput is a performance indicator of interest when designing a CPS. Analyzing throughput reachable by a CPS at design-time implies to optimize the behaviour of the system in such a way that it may run with an optimal frequency. This can be achieved by using synchronous dataflow graphs (SDFGs) which is a formal model of computation that fosters the analysis of systems where performance is always prominent. In this paper, we discuss on the throughput estimation for CPS applications modeled with the SDFGs. In order to evaluate the optimal throughput reachable by a CPS application, we use SDFGs to describe computations and communications in the CPS application and we propose a mathematical formulation of scheduling and mapping decisions in order to deploy the behavioural model of the CPS onto a platform, which essentially consists of heterogeneous and distributed resources.

Keywords: Cyber Physical System· Throughput · Synchronous dataflow.

1 Introduction

Cyber-physical systems (CPSs) are distributed systems consisting of parallel and heterogeneous components (sensors, controllers, actuators.) deeply intertwined and communicating with each other to sense, to control and to execute physical processes [10]. In these systems, performance is usually important since their components execute tasks constrained by timing requirements such as latency and throughput. In order to obtain a valid and implementable CPS, performance analysis is then crucial in the early design-time of the system. To achieve this goal, synchronous dataflow graphs (SDFGs) can be very beneficial. SDFG is a formal model introduced in [17] and widely used to describe communications in embedded and distributed systems and to perform the static analysis of their performance. This model is known to be an equivalent of Weighted Event Graph (WEG) [16], a subclass of Petri Nets which is a general-purpose modeling language often used to model and analyze the timing behaviour of the automated production systems [6]. For the rest of the paper, we adopt the notation SDFG instead of WEG. In this paper, SDFG is used to tackle the static analysis of throughput metrics for CPSs. Analyzing the throughput reachable by a CPS

at design-time means to evaluate the maximum execution frequency of its application over its platform. This can be achieved by scheduling and mapping the application graph of the CPS (i.e the SDFG that models the CPS) to the platform in order to determine when and where the tasks of the application are executed. This paper tackle the throughput analysis problem for a CPS where communications are modeled with SDFG by taking into account the heterogeneity of the CPS resources. In this paper, resources heterogeneity means that there is a potential redundancy of CPS components that may provide the same services but they do not offer the same quality of service (QoS). In this context, the QoS indicator related to the throughput analysis of a CPS is the execution time of CPS tasks.

Various analysis techniques have been proposed previously in the literature of dataflow models and similar tasks model to schedule and evaluate the throughput for applications whose tasks are executed by parallel and distributed resources.

In [4, 6, 9], cyclic schedules were used to characterize and to evaluate throughput for dataflow-based applications. However, the throughput analysis in these paper do not deal with resources constraints of the platform on which the applications are scheduled and mapped.

In [8, 11, 14, 15], various static scheduling techniques have been used to maximize the throughput of dataflow-based applications. Although these techniques take into account some resource constraints, the resources on which the tasks are scheduled and mapped are fully homogeneous.

The scheduling of tasks onto heterogeneous platforms have been addressed in [1, 13]. However, scheduling has been only studied for directed acyclic graph structures. This means that approaches proposed in these papers do not fit for scheduling a task graph that contains cycles.

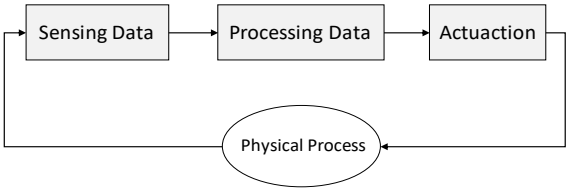
To the best of our knowledge, the current literature of dataflow models do not deal with the scheduling and mapping of SDFGs which contains cycles onto heterogeneous and parallel resources. In this paper, we aim at formulating this scheduling and mapping problem for the CPS by proposing some mathematical models that describe the CPS platform and its tasks graph as well as its scheduling and mapping constraints. Different research directions have also been highlighted to solve the problem.

The rest of the paper is organized as follows. Section 2 is devoted to a detailed description and modeling of a CPS. Next, in section 3 we depict a mathematical model that describe the throughput problem for a CPS and we propose a resolution approach to evaluate the optimal throughput reachable by a CPS. Finally, we draw some conclusions in the section 4.

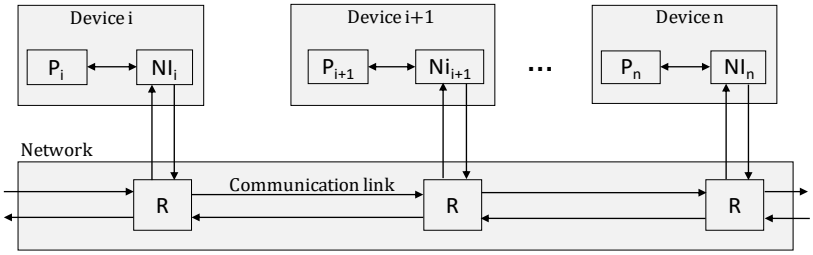
2 Description and Modeling of the CPS

2.1 Architectural description of the CPS

The CPS tackled in this paper consists of a logical part and a physical part. The logical part is an application involving computation functions to sense, control



(a) Coarse view of the CPS function



(b) Overview of the CPS platform.

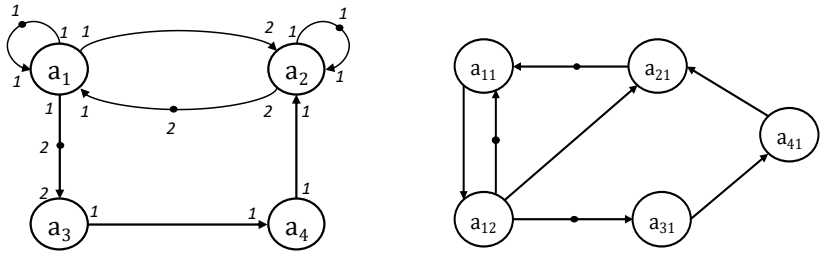
Fig. 1: Architectural Description of the CPS

and execute the physical processes. These functions exchange flows of data to determine the behaviour of the system which mostly act as a loop-control (see Fig. 1a). The physical part of the CPS (see Fig. 1b) is platform including parallel and heterogeneous devices (also called resources) such as sensors, controllers and actuators that interact with each other through a distributed network to process the CPS functions. Each device contains a processor (P) which computes the CPS tasks and a network interface (NI) through which it communicates with the other devices. The network interface is needed to decouples computations from communications and to connect the device with the distributed network. Which network connects all devices via its routers (R) and communication links. The CPS platform provides a resource sharing mechanism that allows several CPS functions to use the communication and computation resources simultaneously while guarantees can be provided on the amount of time a function has access to the resources and frequency of these accesses.

2.2 CPS application model

In order to evaluate the throughput reachable by the CPS, computations and communications are described using SDFG [17]. Indeed, SDFG is a model of computation for a data-driven style of control. Formally, a SDFG is described as a directed graph $G_{sdf} = (\mathcal{A}, E, \mathcal{P}, \mathcal{C}, M_0)$ where:

– $\mathcal{A}$ is the set of actors. Actors are nodes that model the computational functions of the CPS.



(a) CPS application graph (b) Equivalent LCG

Fig. 2: An example of a CPS application.

- E is the set of arcs (also called channels) each modeling the communication between two computational functions.
- $\mathcal{P} = \{p(e)|e \in E\}$ is the set of production rates determined by the function $p : E \to \mathbb{N}*$ which associates to each channel $e \in E$ a fixed number $p(e) = p_e$ indicating the quantity of produced data over the channel when the input function of e is fired.
- $C = \{c(e)|e \in E\}$ is the set of consumption rates. It is determined by the function $c : E \to \mathbb{N}*$ which associates to each channel $e \in E$ a fixed number $c(e) = c_e$ indicating the quantity of consumed data from the channel to fire the output function of e.
- $M_0 = \{M(0)(e)|e \in E\}$ is the initial marking of the graph. The initial marking represents the quantity of data upon the channels at the beginning of a process. It is generally represented by dots also called tokens or delays and is determined by the function $M : E \to \mathbb{N}$ which associates to each channel $e \in E$ a non-negative integer $M_0(e)$ which provides a number of tokens per channel.

Fig. 2a depicts the example of an application graph that models communications between three computational functions of a given CPS. In this application graph, channels are initialized with fixed number of tokens. To analyze the timing behaviour of such an application graph, static scheduling and mapping strategies can be used to determine when and where the actors are executed. However, before scheduling and mapping such a graph to a platform, some useful properties such as consistency and liveness need to be checked. Indeed these properties are the necessary and sufficient conditions that determine the schedulability of any SDFG. When a SDFG is consistent and live it is proved in [17] that there exists a periodic admissible sequential (or parallel) schedule where its actors can be fired infinitely often with a bounded number of tokens.

Definition 1 (Consistency and Repetition Vector). *Let consider a SDFG* $G_{sdf} = (\mathcal{A}, E, \mathcal{P}, \mathcal{C}, M_0)$. G_{sdf} *is consistent if there exists a function* $n : E \to \mathbb{N}*$ *which associates to each actor, a strictly positive integer such that for any channel* $e = (a_i, a_j, p_e, c_e, M_0(e)) \in E$, $p_e.n(a_i) = c_e.n(a_j)$. *The set of values provided*

by such function determines the repetition vector $N = [n(a_1), ..., n(a_{|A|})]$ *associated with* G_{sdf} *where* $n(a_i)$ *is the number of times the actor fires within a single execution cycle of* G_{sdf}.

According to Definition. 1, the application graph showed in Fig .2a is consistent and its repetition vector is given by N=[2,1,1,1]. According to this vector, the actor a_1, fires twice while the other actor fires once in a single iteration of the graph.

Definition 2 (Liveness). *Let* $G_{sdf} = (A, E)$ *be a SDF model where A is a finite set of actors, E is a finite set of channels.* G_{sdf} *is live if and only if its initial marking enables to execute the SDF actors infinitely often without deadlocks.*

Many algorithms have been proposed to check the liveness of SDFGs [7, 12]. In this paper, we assume that any SDFG that models a CPS application is always consistent, live and then schedulable.

2.3 From a CPS application graph to a Linear Constraint Graph

Before scheduling a SDFG on a parallel platform, it is important to capture all the dependencies between the firings of actors. To achieve this, SDFGs are often transformed into precedence constraint graphs. There are three main approaches to transform a SDFG into a precedence constraint graph. The first approach is based on the transformation of the SDFG into an equivalent Homogeneous Synchronous Dataflow Graph (HSDFG), which is a SDFG where all the input and output rates are equals to one [17, 8, 4]. The second approach is by exploring the state-space of a simulated self-time execution of the SDFG until a periodic phase is found [11]. Such simulation-based method avoids the transformation from SDF into HSDFG. The third approach is to transform the SDFG into a linear constraint graph (LCG) which is a smaller sub-graph of a HSDFG [5]. These three approaches were implemented and compared in [5] and experimental results showed that the third approach is more efficient than the first two. Therefore, we use the algorithm proposed in [5] to transform any application graph of a CPS into an equivalent LCG.

According to the algorithm proposed in [5], the corresponding LCG for the SDFG depicted in Fig .2a is depicted in Fig .2b. In the LCG, nodes characterize the firings of actors belonging to the original SDFG and arcs characterize the precedence relations between these firings. Dots on the arcs (a_{12}, a_{11}), (a_{21}, a_{11}) and (a_{12}, a_{31}) indicate that the downstream actors (i.e a_{11} and a_{31}) are fired in the next iteration of the LCG. In general, a dot on an arc xy means that the corresponding token is produced in one iteration and consumed in the next iteration. Note that in an LCG, an arc can contain at most one token.

2.4 CPS platform model

Before analyzing the throughput for the CPS, we also need to provide a formal description of the platform on which the CPS application is executed.

The targeted platform consists of n heterogeneous devices r_u, $1 \leq u \leq n$ fully interconnected as a virtual clique by m heterogeneous communication medium c_i, $1 \leq i \leq m$. A communication medium is a bidirectional link $L_{u,v} : r_u \rightarrow r_v$ between any pair of device r_u and r_v, of data transmission time δ_{r_u,r_v}. Note that the heterogeneity of devices means that they may execute the same actors with different execution times while the heterogeneity of the communication medium means that the communication times are different between all the devices. Each device may process one or several actors. A set of tuples $(a_k, \ell_{r_u,a_k}), 1 \leq k \leq p$ is associated with each device r_u to specify the set of actors a_k that it executes as well as the corresponding execution times ℓ_{r_u,a_k}. More succinctly a platform model is defined as follows:

$$\{r_u = \{(a_k, \ell_{r_u,a_k})\}\} \cup \{c_i = (L_{u,v}, \delta_{r_u,r_v})\}$$

An instance of platform model for the application graph depicted in Fig .2a, is given by:

$\{r_1 = \{(a_1, 8), (a_2, 4)\}; r_2 = \{(a_2, 5), (a_3, 6)\}; r_3 = \{(a_2, 8)\}\} \cup \{c_1 = (L_{r_1,r_2}, 5);$
$c_2 = (L_{r_1,r_3}, 3); c_3 = (L_{r_2,r_3}, 2)\}$

3 Throughput evaluation

Determining the optimal throughput reachable by the application graph of a CPS implies to schedule and map the actors of its equivalent LCG to the platform in such a way that the iteration period of the LCG can be minimized. In order to achieve this goal, there is a need of defining a decision model that express the scheduling and mapping constraints as well as the objective function that need to be optimized when scheduling and mapping the LCG to the CPS platform.

3.1 Scheduling and Mapping Decision Model

Let G_{sdf} be a SDFG, G_{lcg} be its equivalent LCG, P_G be the platform on which G_{lcg} is scheduled and mapped and S be the function that schedules and maps actors of G_{lcg} to P_G. The throughput λ of G_{sdf} is defined as the average iteration number of G_{lcg} per time units in S. G_{lcg} achieves a single iteration if all its nodes are scheduled and mapped onto the P_G. If T is the iteration period associated with G_{lcg} then the throughput λ is measured as the inverse of the iteration period (i.e $\lambda = 1/T$). Therefore, maximizing the throughput of an SDFG implies to minimize the iteration period of its equivalent LCG. In order to achieve this, we formulate the following mathematical model to describe the scheduling and mapping problem for any LCG. Note that in the proposed model, an actor is considered as a node in the LCG.

Variables

- $\boldsymbol{S_{a_i}}$: the starting time of an actor $\boldsymbol{a_i}$ in a schedule $\boldsymbol{S}$.
- $\boldsymbol{X_{r_u,a_i}}$: a binary variable set to 1 if the actor $\boldsymbol{a_i}$ is mapped onto the device $\boldsymbol{r_u}$ and 0 otherwise.
- $\boldsymbol{Y}$: the minimal iteration period.

Model

$$
\begin{cases}
f_{objective} = min\ Y \\
\sum_{u=1}^{n} X_{r_u,a_i} = 1, \quad \forall\ a_i. & (1) \\
S_{a_i} + \sum_{u=1}^{n} X_{r_u,a_i} \cdot \ell_{r_u,a_i} \leq S_{a_j},\ \forall a_i \prec a_j. & (2) \\
S_{a_i} + \sum_{u=1}^{n} X_{r_u,a_i} \cdot \ell_{r_u,a_i} \leq Y, \quad \forall\ a_i. & (3) \\
(X_{r_u,a_i} \cdot X_{r_u,a_j} \cdot (S_{a_i} + \ell_{r_u,a_i}) \leq S_{a_j}) \vee (X_{r_u,a_i} \cdot X_{r_u,a_j} \cdot (S_{a_j} + \ell_{r_u,a_j}) \leq S_{a_i}),\ \forall\ a_i, a_j. & (4) \\
X_{r_u,a_i} \in \{0,1\},\ \forall r \in \mathcal{R} & (5) \\
S_{a_i} \geq 0 & (6) \\
Y \geq 0 & (7)
\end{cases}
$$

The objective function of this model is to map the tasks onto resources of the platform and order their executions so that the constraints are satisfied and a minimum overall completion time is obtained.

Constraint (1) states that any actor of the LCG must be mapped on exactly one CPS device. Constraint (2) depicts the precedence constraint between two dependent tasks in the LCG. Indeed, for any actor a_i and a_j if there exists a precedence relation from a_i to a_j (i.e $a_i \prec a_j$), and if these actors are respectively executed over the resources r_u and r_v then, the starting time of a_j is greater or equal to the sum of starting and processing time of a_i in device r_u. For an iteration of the LCG, the constraint (3) states that the finishing time of each actor is lower of equal to the minimal iteration period. Constraint (4) states the restriction that if any actor a_i (or a_j) is first assigned to resource r_u, no other actor a_j (a_i) can start onto the resource until a_i (or a_j) is completely processed. Constraints (5), (6) and (7) define the range of optimization variables.

3.2 Resolution Method for Mapping and Throughput Computation

To solve this mapping problem and evaluate the maximum throughput, we split it into two different sub-problems.

Assuming that for any SDFG, the equivalent LCG does not contain cycles. In this case, the mapping problem refers to a well-known NP-complete problem which is mapping a DAG to a parallel and distributed platform under communication constraints. Several approaches have been proposed to tackle this kind of problem. In [1, 13], differents list-based heuristics have been proposed that provide mapping solutions with reasonable computation times to solve this problem. However, in this paper, as we assume that a LCG always contains cycles (e.g. see Fig. 2b), we need to reduce the LCG structure into a DAG structure without losing the semantic of the initial model. Afterwards, the DAG can be scheduled and mapped to the heterogeneous platform using list-scheduling heuristics. Finally, according to the resulted schedule, the throughput of the CPS application graph can be evaluated.

A useful technique has been proposed in [3] to transform a graph that contains cycles into DAG and to map it on an homogeneous platform consisting of two processors. This transformation technique can be extended and used in the context of our study to transform a LCG into a DAG. After transforming the LCG into DAG, existing heuristics [1, 13] for mapping a DAG to a heterogeneous platform should be implemented and compared by running them over various instances of LCG in order to determine the ones that provides the optimal throughput reachable by the application graph of the CPS.

4 Conclusion and Outlooks

In this paper, we discuss the throughput evaluation of a CPS modeled with SDFG. First we provide a formal description of the CPS platform and application. Then, we formulate the scheduling and mapping problem for maximizing the throughput of a CPS application considering some resources constraints related to the CPS platform. Today, the problem is not solved yet, however, research directions have been highlighted to tackle it. As first perspective, we plan to propose an efficient algorithm to transform the precedence constraint graph (i.e. the LCG) of a CPS application into a DAG that preserves the semantics of the original graph. Afterwards, we plan to design and implement an efficient list schedule heuristic for scheduling and mapping the DAG to the CPS platform in order to evaluate the maximal throughput reachable by the CPS application.

References

1. A. Emeretlis, T. Tsakoulis, G. Theodoridis, P. Alefragis and N. Voros, "Task graph mapping and scheduling on heterogeneous architectures under communication constraints," 2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), Pythagorion, 2017, pp. 239-244.
2. Youen Lesparre. Efficient evaluation of mappings of dataflow applications onto distributed memory architectures. Mobile Computing. University Pierre et Marie Curie - Paris VI, 2017. English.

3. W.N.M Ariffin, "Task Scheduling for Directed Cyclic Graph Using Matching Technique," Contemporary Engineering Sciences, Vol. 8, 2015, no. 17, 773 -788, HIKARI Ltd.
4. Bodin B., Munier-Kordon A., and De Dinechin, B. D. (2012).,K-periodic schedules for evaluating the maximum throughput of a synchronous dataflow graph. In 2012 International Conference on Embedded Computer Systems (SAMOS), pages 152159.
5. De Groote, R., Kuper, J., Broersma, H., and Smit, G. J. (2012). Max-plus algebraic throughput analysis of synchronous dataflow graphs. In 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pages 2938. IEEE.
6. Benabid-Najjar, A., Hanen, C., Marchetti, O., and Munier-Kordon, A. (2012). Periodic schedules for bounded timed weighted event graphs. IEEE Transactions on Automatic Control, 57(5):12221232.
7. O. Marchetti et A. Munier Kordon A sufficient condition for the liveness of weighted event graphs European Journal of Operational Research,197(2), pp. 532-540, Sept 2009.
8. S. Sriram and S. S. Bhattacharyya, Embedded Multiprocessors: Scheduling and Synchronization, 2nd ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.
9. C. Hanen : Cyclic scheduling, chapter in Introduction to Scheduling, Y. Robert, F. Vivien (Eds.), pp. 103-128, (Chapman and Hall/CRC Computational Science), (ISBN: 978-1420072730) (2009).
10. E. A. Lee, "Cyber Physical Systems: Design Challenges," 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), Orlando, FL, 2008, pp. 363-369.
11. A. Ghamarian, M. Geilen, S. Stuijk, T. Basten, A. Moonen, M. Bekooij, B. Theelen, and M. Mousavi. Throughput analysis of synchronous data flow graphs. In ACSD06, Proc. (2006), IEEE.
12. A. H. Ghamarian, M. C. W. Geilen, T. Basten, B. D. Theelen, M. R. Mousavi and S. Stuijk, "Liveness and Boundedness of Synchronous Data Flow Graphs," 2006 Formal Methods in Computer Aided Design, San Jose, CA, 2006, pp. 68-75.
13. Topcuoglu, Haluk; Hariri, Salim; Wu, M. (2002). "Performance-effective and low-complexity task scheduling for heterogeneous computing". IEEE Transactions on Parallel and Distributed Systems. 13 (3): 260274.
14. P.-Y. Calland, A. Darte, and Y. Robert. Circuit retiming applied to decomposed software pipelining. IEEE Transactions on Paralllel and Distributed Systems, 9(1):2435, 1998.
15. F. Gasperoni and U. Schwiegelshohn. Generating close to optimum loop schedules on parallel processors. Parallel Processing Letters, 4:391403, 1994.
16. Teruel, E., Chrzastowski-Wachtel, P., Colom, J., Silva, M.: On weighted t-systems. In: Application and Theory of Petri Nets 1992, pp. 348367 (1992).
17. E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," in Proceedings of the IEEE, vol. 75, no. 9, pp. 1235-1245, Sept. 1987.