Noname manuscript No.
(will be inserted by the editor)

Mathematical Programming Models for Scheduling in a CPU/FPGA Architecture with Heterogeneous Communication delays

Abdessamad Ait El Cadi · Omar Souissi · Rabie Ben Atitallah · Nicolas Belanger · Abdelhakim Artiba

Received: date / Accepted: date

Abstract This paper deals with the mathematical modelling of a scheduling problem in a heterogeneous CPU/FPGA architecture with heterogeneous Communication delays in order to minimize the makespan, C_{max}. This study was motivated by the quality of the available solvers for Mixed Integer Program. The proposed model includes the Communication delays constraints in a heterogeneous case, depending on both tasks and computing units. These constraints are linearized without adding any extra variables and the obtained linear model is reduced to speed-up the solving with CPLEX up to 60 times. Computational results show that the proposed model is promising. For an average sized problem up to 50 tasks and 5 computing units the solving time under CPLEX is about few seconds.

Keywords Heterogeneous System · Task Scheduling · CPU/FPGA · Communication delays · Mixed Integer Program · CPLEX

1 Introduction

Real-time computing systems are increasingly used in several industrial domains such as aerospace, avionic, rail, and automotive. During the manufacturing process, designers need development tools for the Verification and the Validation (V&V) of modern and complex systems. Today, the simulation phase is considered as an unavoidable part of the V&V cycle. In order to meet the application requirements in terms of increasing computation rate and real-time, the behavior of the system has to be validated first at the simulation level before integrating the functionality into the real system.

A. Ait El Cadi · O. Souissi · R. Ben Atitallah · N. Belanger · A. Artiba
UVHC - LAMIH CNRS UMR 8201
F-59313 Valenciennes Cedex 9
Tel.: +33-3-27-51-19-46
Fax: +33-3-27-51-13-16
E-mail: Abdessamad.AitElCadi@univ-valenciennes.fr

Due to the ever-changing face of technology, we lead in collaboration with Airbus Helicopters[1] research department the development of innovative simulation systems. The objective of this project is to bring reliability and competitiveness to the avionic industry. The efficient way to achieve a real-time capabilities and a high level of performance needed for these simulations is to distribute computations over a cluster of heterogeneous architectures (see [1]) with a combination of general CPUs (Central Processing Unit) and reconfigurable fabrics provided by standard FPGAs (Field-Programmable Gate Array) circuit. In such systems, the multi-core CPUs provide raw computation rates and ease of programming while the reconfigurable logic offers high performance per watt and adaptability to the application constraints.

Designers could exploit the existing partitioning in the application (i.e. hardware-software and parallel-sequential hardware) which leads to several feasible implementations whose performances vary with the chosen partitioning. With the management of the parallelism intrinsic in the application, FPGA technology could offer better performances comparing to CPUs up to 10x [1] at lower frequencies. Using heterogeneous CPU/FPGA systems allows to adapt the architecture according to the application constraints and thus to optimize hardware resources. All these benefits emphasize hardware designers to redirect their efforts on CPU/FPGA architecture. This heterogeneous architecture offers the ability to address specific application constraints (timing deadlines, power consumption, etc.) and leads to get the maximum benefits of the performance of each part of the designed system.

The main challenge faced by developers using these architectures relates to the mapping of the application tasks onto these resources (CPU and FPGA). They need methods and tools that help to perform this mapping efficiently while considering real-time constraints. This challenge involves the static and the run-time task mapping. As highlighted in [2], the problem is how to assign tasks to the available resources in order to optimize some performance criterion such as the makespan, the load balance, etc.

From the computing point of view and considering that processors will carry out various tasks with different data, our case is of MIMD (Multiple Instructions Multiple Data) architecture according to Flynn's taxonomy (which is a classification of computer architectures, based on the number of streams of instructions and data) [4]. From the operation research view and according to El-Rewini [5], who made a classification of different scheduling problems, we could say that our research focuses on scheduling problem with precedence and heterogeneous communication delays. In this paper, we consider the deterministic case, where the task graph topology, the execution and communication costs are known prior to execution.

Several researchers consider scheduling and load balancing problems in different areas [8,18] such as flow-shop [20], job-shop [10,22], scheduling in parallel and distributed systems [13,3], resource-constrained project problem

[1] Airbus Helicopters is the leader in civil and military helicopter manufacturing. http://www.airbushelicopters.com

[9,21], [15,24], assembly line balancing problem [16], [17], process planning [26, 27], and simulation in a distributed environment [12,3].

It is well known that the most of classical scheduling problems are NP-hard problems ([6] and [14]). Moreover, in practice the scheduling problems are harder since they incorporate additional side constraints and / or optimize more than one objective. In this paper, we propose a mathematical formulation for our scheduling problem with the communication constraints. Then we propose a linear formulation and a preprocessing step to reduce the size of this model.

Contributions. In this paper the following contributions may be observed: (i) A mathematical model for an heterogeneous computing architecture CPU/FPGA with a heterogeneous Communication delays; (ii) We linearize the model without adding any extra variables; (iii) We reduce the size of the model by eliminating some disjunctive constraints. This leads to a model that may contain only $O(n)$ variables for a fixed number of computing units in the case of graphs with high density, greater than 50%; (iv) Extensive comparative analyses on generated simulation project instances are performed. They confirm the quality of the reduced model.

Outline. This paper is organized as follows. Section 2 introduces the problem of the "Task Scheduling in a CPU/FPGA Architecture" illustrated by an example. Section 3 gives an overview of the related works. Our mathematical models are exposed in Section 4. In Section 5, we present several results on different instances of our scheduling problem. Then we conclude this work in Section 6.

2 Problem description

The problem is to map efficiently the application (simulation project) tasks onto a heterogeneous CPU/FPGA architecture. Figure 1 gives an overview of the problem. The upper part of this figure shows the task graph of the simulation project to separate. The middle part illustrates the heterogeneous network of computing units on which the application should be mapped. The lower part of this figure gives an example of a solution to the presented problem.

A task is an abstract object that corresponds to a non-empty set of instructions, and that has the following attributes: name, identification number, type of implementation and the number of elementary instructions. There are four types of implementation:

- Type 1: software exclusive task; it should be planned only on a CPU.
- Type 2: hardware exclusive task; it should be planned only on a FPGA.
- Type 3: hardware or software exclusive task; it could be planned on a FPGA or on a CPU but not splitted between them.
- Type 4: hardware or software splittable task; it could be planned on a FPGA or on a CPU or splitted between them.

The presented problem in the upper part of the Figure 1 contains 5 tasks to be executed on 3 CPUs and 1 FPGA. The first table in this upper part gives

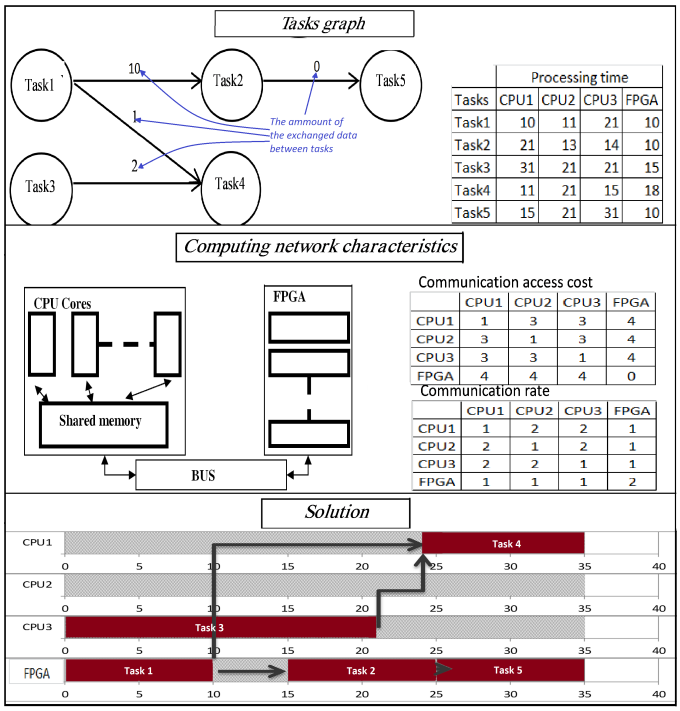

Fig. 1 General overview of the problem

the processing times of each task depending on the chosen computing unit. Precedence structure given in the tasks graph indicates that task 1 and task 3 has no predecessors, while tasks 4 and 5 have no successors. The edge weight indicates the amount of exchanged data. In this example, the link from task 1 to task 2 indicates that task 2 needs a data of size 10 to be received from task 1, before starting the execution. The time cost to exchange this amount of data depends on the link between the processing units to whom the tasks were assigned.

As shown in the middle part of Figure 1, the heterogeneous computing system is composed by a set of cores and a set of FPGAs. They are linked by different communication mediums (In Figure 1, the CPU cores are linked by a shared memory and the FPGA communicates with the CPU cores via a bus). The two tables in this middle part, give the communication rate (number of exchanged data unit per unit of time) between the processing units and the access cost, the fixed amount of time needed to start communication. This network impacts the scheduling problem. For instance, if two tasks are related by a precedence constraint, the communication delay between their execution will depend on the link between the two processing units, to which they are assigned. To clarify this point, the lower part of Figure 1 presents a solution of this problem as a Gantt diagram. We highlight in this diagram the communication effect on the scheduling solution. We could notice in this

diagram, that the delay to send the 10 data units from task 1 to task 2 on the FPGA is $0 + 0.5 \times 10 = 5$ time units.

The scheduling and load balancing problem is a very generic problem, which involves several fields such as: industry, project management and computer science. The problem is to assign each operation (task) to a machine (CPU or FPGA) and to sequence the operations on the machines (CPUs/FPGAs), such that a criterion is optimized, for instance the maximal completion time (makespan) of all operations (tasks) is minimized. More precisely, we have to assign a set of tasks to a set of heterogeneous processing units such that each task respects the precedence and communication constraints to guarantee that the semantics of the original program are preserved. Therefore, for each task, we need to decide the time at which its execution will start and the processing unit that is responsible for its execution. Each task must be executed and the whole computation project should be planned on the available resources. Note that a task can be executed on a given processor only when all data from its predecessors become available to that processor. There is no overlapping between tasks assigned to the same processing unit. Each processing unit could execute only one task at a time. The preemption is not allowed in our case. In practice, there are also some preferences to take into account such as some tasks could be only executed on a certain group of resources (see type of implementation of a task).

3 Related works

Rayward-Smith [38] was the first to address the Communication delays case in solving Multiprocessors Scheduling Problem (MSP). He studied the case with a unit Communication delays, prove that it is NP-hard and propose a generalized list schedule. Chrétienne and Picouleau [25] did a survey for the general case with communication delays. They presented a list of existing variants on Multiprocessors Scheduling Problem with Communication Delays (MSPCD) and give an overview of their complexity and solution methods. For each class, they described the current knowledge regarding the edge between easy and difficult problems.

The majority of the related works deals with a meta-heuristic based solution techniques, since the MSPCD problems are NP-hard (see e.g. [6,14]). Moreover, the practical scheduling problems are even harder since they incorporate additional side constraints and/or optimize more than one objective. Hwang et al. [32] present a comparison of algorithms for the case of identical processors with communication cost. Luo et al. [35] review 20 greedy algorithms for cases without communication. The m-machines problem with communication delays has been proven to be NP-complete for arbitrary precedence relations even for two processors in [37]. Nevertheless, there are some special classes for which the polynomial time algorithms may be constructed [19,33,37].

Even though, prior to developing an effective heuristic, researchers usually start with a mathematical programming formulation of the scheduling problem under study. Although several mathematical programming models have been developed, they typically do not perform very well for practically motivated problem instances due to model formulation and/or computational difficulties [40]. Unlu and Mason [40] give an overview of the different mathematical programming formulations for the case without communication. They document four different Mixed Integer Programming (MIP) formulations based on four different types of decision variables: Assignment and positional date variables [28], Linear ordering variables [36], Time indexed variables [39] and Network variables [23]. We enumerate in our review three approaches that take into account the communication delays: Davidović et al. [29,30] and Venugopalan and Sinnen [41]. In these approaches, the suggested linearization increases the size of the problem significantly. For example, linearized models contained variables with up to 4 indices. We also noted that the introduced variables allow some events to be over defined. For example, one set of variables assigns tasks to processors and another set is used to store for each task the assigned processor index. It is clear that the same information is coded twice.

4 Mathematical models

In this section, we aim to build a mathematical model for our scheduling problem. Initially it deals with a simulation project – formed by several tasks – to separate and to schedule within a given number of CPUs and FPGAs. We propose here to model the problem described in Section 2. The notations are described as follows:

- N: Set of n tasks;
- M: Set of m processing units (CPUs/FPGAs);
- $G = (N, A)$: a given directed acyclic graph, where N is the set of tasks and A is the set of arcs representing the precedence between tasks, i.e. (i, j) in A means that task i must be performed before the task j.
- $Pred(i)$: Set of tasks that precede task i;
- t_{ik}: Processing time of task i on processing unit k;
- $c_{ik,jl}$: The cost of direct communication between task i on processing unit k and the task j on processing unit l;
- F_k : Set of tasks that should not be assigned to the processing unit k.

Communications depend on the architecture of the hardware used, such as shared memory, Ethernet links, and so on. The network architecture (for example full connected, ring, hypercube and star) also has an impact on the communication. Generally, the communication cost is nonlinear, the communication cost function is defined as $c_{ik,jl} = a_{kl} + \frac{d_{ij}}{r_{kl}}$, where a_{kl} is the fixed communication cost between processing units k and l, d_{ij} is the size of data sent from task i to task j, and r_{kl} is the communication rate between the two processors k and l. We assume that $r_{kl} = r_{lk}$ and $a_{kl} = a_{lk}$.

4.1 Non-linear mathematical model

There are many examples of an integer linear program models for solving scheduling problems. But they do not take into account all the constraints. For example, in assembly line balancing, Urban [16] build a model that minimizes a number of used machines with a respect of precedence constraint. But this model could not fit to our communication problem. In parallel computing, Darte [7] gives another linear program but with no respect of communication constraints. Also Chen and Lin [11] make a non-linear program which minimizes the communication costs under a computing capacity constraints.

In this section, we propose a non-linear model with the communication delay. For the decision variables, we use assignment and positional date variables:

$$x_{ik} = \begin{cases} 1 & \text{if task } i \text{ is assigned to the processor } k; \\ 0 & \text{else.} \end{cases}$$

s_i - the starting time of task i.

We choose these variables since this choice reduces the number of variables, used in [29] and [41].

The mathematical model can be formulated as follows:

$$\min C_{max} \tag{1}$$

subject to:

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad \forall i \in N \tag{2}$$

$$s_i + \sum_{k=1}^{m} t_{ik} x_{ik} \leq C_{max} \qquad \forall i \in N \tag{3}$$

$$s_i + \sum_{k=1}^{m} t_{ik} x_{ik} + \sum_{k=1}^{m} \sum_{l=1}^{m} c_{ik,jl} x_{jl} x_{ik} \leq s_j \ \forall j \in N, \forall i \in Pred(j) \tag{4}$$

$$\begin{cases} s_i + t_{ik} - s_j \leq B(1 - x_{ik}x_{jk}) \\ \qquad or \qquad\qquad\qquad \forall i, j \in N, \forall k \in M \\ s_j + t_{jk} - s_i \leq B(1 - x_{ik}x_{jk}) \end{cases} \tag{5}$$

$$x_{ik} = 0 \quad \forall i \in F_k, \forall k \in M \tag{6}$$

$$x_{ik} \in \{0, 1\}; s_i \in R^+ \quad \forall i \in N, \forall k \in M \tag{7}$$

The objective function (1) minimizes the makespan. The set of constraints (2) means that each task must be assigned to one and only one processing unit. The set of constraints (3) expresses that the C_{max} (the makespan) corresponds to the total length of the schedule (i.e. the maximum of time when the tasks have finished processing). The set of precedence constraints (4) describes that each task j which succeeds a task i must be carried out after the starting time of task i, plus the processing time of its task and time of communications.

The set of disjunctive constraints (5) asserts that two tasks assigned to the same processing unit must not overlap. In these constraints we use B, a big value such that the constraints hold only for the tasks assigned to the same computing unit, i.e. $x_{ik} = x_{jk} = 1$. The set of constraints (6) is aimed to avoid the forbidden assignments.

4.2 A linear mathematical model

The constraints (4) and (5) are quadratic but the objective function and all the other constraints are linear which made our program as Mixed Integer Quadratic Constrained Program (MIQCP). In this section we propose a linear mixed integer program.

The communication and precedence constraints (4) could be linearized, in our case, without adding any extra variables. These constraints are equivalent to:

$$s_i + t_{ik}x_{ik} + c_{ik,jl}(x_{jl} + x_{ik} - 1) \leq s_j \quad \forall k, l \in M; \forall j \in N, \forall i \in Pred(j) \qquad (4a)$$

To justify the equivalence between the constraints (4) and (4a), let i and j two tasks such as i is a predecessor of j, i.e. $i \in Pred(j)$. Since each task $i \in N$ is assigned to exactly one and only one processor CPU or FPGA (i.e. $\sum_{k=1}^{m} x_{ik} = 1$), the two sums $u = \sum_{k=1}^{m} t_{ik}x_{ik}$ and $w = \sum_{k=1}^{m}\sum_{l=1}^{m} c_{ik,jl}x_{jl}x_{ik}$ contain only one term different from zero. In fact $u = t_{ik^*}$ and $w = \sum_{k=1}^{m} x_{ik}(\sum_{l=1}^{m} c_{ik,jl}x_{jl}) = c_{ik^*,jl^*}$, where $x_{ik^*} = x_{jl^*} = 1$. Hence, the constraints (4) are equivalent to the following constraints:

$$s_i + t_{ik}x_{ik} + c_{ik,jl}x_{jl}x_{ik} \leq s_j, \forall k, l \in M \quad (4b)$$

The preceding constraints (4b) can be explicitly described by one of the following three constraints:

$$\begin{aligned} s_i + t_{ik^*} + c_{ik^*,jl^*} &\leq s_j, \\ s_i + t_{ik^*} &\leq s_j, \\ s_i &\leq s_j. \end{aligned}$$

These constraints are dominated by the following constraint:

$$s_i + t_{ik^*} + c_{ik^*,jl^*} \leq s_j.$$

Moreover, the constraints (4a) can be explicitly described by one of the following four constraints:

$$\begin{aligned} s_i + t_{ik^*} + c_{ik^*,jl^*} &\leq s_j, \\ s_i + t_{ik^*} &\leq s_j, \\ s_i &\leq s_j, \\ s_i - c_{ik,jl} &\leq s_j. \end{aligned}$$

Again, these preceding constraints are dominated by the following constraint:

$$s_i + t_{ik^*} + c_{ik^*,jl^*} \leq s_j.$$

This justifies the linearization of constraints (4) by (4a).

The disjunctive constraints (5) could be linearized in two steps. First we replace $B(1 - x_{ik}x_{jk})$ by $B(2 - x_{ik} - x_{jk})$:

$$(5) \leftrightarrow \begin{cases} s_i + t_{ik} - s_j \leq B(2 - x_{ik} - x_{jk}) \\ \quad or \qquad\qquad\qquad\qquad \forall i, j \in N, \forall k \in M. \\ s_j + t_{jk} - s_i \leq B(2 - x_{ik} - x_{jk}) \end{cases}$$

Second, we introduce another set of binary decision variables δ_{ij}. These new variables could be seen as the decision that either we chose to execute task i before j ($\delta_{ij} = 1$) or the opposite ($\delta_{ij} = 0$). The disjunctive constraints become:

$$(5) \leftrightarrow \begin{cases} s_i + t_{ik} - s_j \leq B(2 - x_{ik} - x_{jk}) + B(1 - \delta_{ij}) \\ \quad and \qquad\qquad\qquad\qquad \forall i, j \in N, \forall k \in M. \\ s_j + t_{jk} - s_i \leq B(2 - x_{ik} - x_{jk}) + B\delta_{ij} \end{cases}$$

$$(5) \leftrightarrow \begin{cases} s_i + t_{ik} - s_j \leq B(3 - x_{ik} - x_{jk} - \delta_{ij}) \\ \quad and \qquad\qquad\qquad\qquad \forall i, j \in N, \forall k \in M. \\ s_j + t_{jk} - s_i \leq B(2 - x_{ik} - x_{jk} + \delta_{ij}) \end{cases}$$

And the model becomes a Mixed Integer Linear Program (MILP). The new model is as follow:

$$\min C_{max} \tag{8}$$

subject to:

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad \forall i \in N \tag{9}$$

$$s_i + \sum_{k=1}^{m} t_{ik}x_{ik} \leq C_{max} \qquad \forall i \in N \tag{10}$$

$$s_i + t_{ik}x_{ik} + c_{ik,jl}(x_{jl} + x_{ik} - 1) \leq s_j \ \forall k, l \in M \forall j \in N, \forall i \in Pred(j) \tag{11}$$

$$s_i + t_{ik} - s_j \leq B(3 - x_{ik} - x_{jk} - \delta_{ij}) \qquad \forall i, j \in N, \forall k \in M \tag{12}$$

$$s_j + t_{jk} - s_i \leq B(2 - x_{ik} - x_{jk} + \delta_{ij}) \qquad \forall i, j \in N, \forall k \in M \tag{13}$$

$$x_{ik} = 0 \qquad \forall i \in F_k, \forall k \in M \tag{14}$$

$$x_{ik} \in \{0, 1\}; s_i \in R^+ \qquad \forall i \in N, \forall k \in M \tag{15}$$

4.3 Reduction of the size of the linear mathematical model

In the previous MILP all the constraints are not necessary. For example, it is not necessary to define the constraints (11) if either task i belongs to the set F_k or task j belongs to the set F_l. Moreover, the constraints (12) and (13) are defined only if the two tasks i and j are not linked with a path of the graph.

To reduce also the number of binary variables we do not use x_{ik} for any i belonging to F_k (i.e. any task i that should not be assigned to the computing

unit k). This will also allow us to remove constraints (14). Furthermore, the variable δ_{ij} will not be used for any pair (i, j) such that i and j belong to the same path.

For the new model we use the following additional notations:

- N^+: Set of tasks with no successors.
- $P(i)$: Set of tasks j such that i and j belong to the same path.

For pruning the model, we compute the set $P(i)$ of tasks j sharing the same path with i in G as a pre-processing phase. It is based on the breadth-first search (BFS) [34] method. Therefore, the reduced linear model is as follows:

$$\min C_{max} \tag{16}$$

subject to:

$$\sum_{k \in \{l \in M / i \in F_l\}} x_{ik} = 1 \qquad \forall i \in N \tag{17}$$

$$s_i + \sum_{k=1}^{m} t_{ik} x_{ik} \leq C_{max} \qquad \forall i \in N^+ \tag{18}$$

$$s_i + t_{ik} x_{ik} + c_{ik,jl}(x_{jl} + x_{ik} - 1) \leq s_j \;\; \forall k, l \in M, \forall j \in N \backslash F_l, \; \forall i \in Pred(j) \backslash F_k \tag{19}$$

$$s_i + t_{ik} - s_j \leq B(3 - x_{ik} - x_{jk} - \delta_{ij}) \;\; \forall k \in M, \forall i \in N \backslash F_k, \; \forall j \in N \backslash (P(i) \cup F_k) \tag{20}$$

$$s_j + t_{jk} - s_i \leq B(2 - x_{ik} - x_{jk} + \delta_{ij}) \;\; \forall k \in M, \forall i \in N \backslash F_k, \; \forall j \in N \backslash (P(i) \cup F_k) \tag{21}$$

$$x_{ik} \in \{0, 1\}; s_i \in R^+ \;\; \forall k \in M, \forall i \in N \backslash F_k \tag{22}$$

To characterize the reduction of the model we need the definition of the set A_H (see [31]). It is the edge set of the complement graph of the undirected graph associated with the n^{th} power G^n of graph G. Where the power G^n of graph G is a graph with the same vertices as G and an arc is drawn between two vertices i and j in G^n if there is a path, of length less or equal to n, from i to j in G.

Now, the size of the Model given by (16)-(22) could be expressed as follows:

$$Number\ of\ variables = nm + n + 2|A_H|$$
$$Number\ of\ constraints = n + |N^+| + m^2|A| + 2|A_H|$$

where $|A_H|$ is the cardinality of A_H.

In fact, in the reduced model, the number of variables and constraints are obviously $nm + n + \sum_{i=1}^{n} |N \backslash P(i)|$ and $n + |N^+| + m^2|A| + \sum_{i=1}^{n} |N \backslash P(i)|$, respectively. In order to get result, we just need to show that $\sum_{i=1}^{n} |N \backslash P(i)| = 2|A_H|$. The sets *P(i)* could be characterized as (see e.g. [31]): $|P(i)| = 1 + d^+_{G^n}(i) + d^-_{G^n}(i), \forall i \in N$, where the $d^+_{G^n}(i)$ and $d^-_{G^n}(i)$ denote, respectively, the out-degree and the in-degree of the vertex i in

the n^{th} power of graph G, i.e. G^n. Let $H=(N,A_H)$ be the complement graph of the undirected graph associated with G^n, where A_H represents the set of the edges in the undirected graph H. Then we have:
$|N \backslash P(i)| = n - (1 + d^+_{G^n}(i) + d^-_{G^n}(i)) = (n-1) - d_{G^n}(i) = d_H(i), \forall i \in N.$ $\diamond$

This result corresponds to the general task graphs. For more specific structure of task graph we derive the following obvious result: If the undirected graph associated with G is a complete graph or if G is a chain, then $P(i) = N, \forall i \in N$. Thus, the number of variables and constraints become $(nm+n)$ and $(2n+m^2|A|)$ respectively.

Generally, if the inverse graph of the undirected graph associated with the n^{th} power graph is totally disconnected, the number of variables is $(nm+n)$ and the problem is easy to solve. Also, the fact that the problem contains disjunctive constraints could be known in advance by calculating $|A_H|$ from the following equation:

$$|A_H| = \sum_{\substack{1 \le i < n \\ i < j \le n}} \left[\neg \left(\bigvee_{i=1}^{n} Adj^i \right) \right] (i,j) = \sum_{\substack{1 \le i < n \\ i < j \le n}} \left[\neg \left((Adj + I)^n \right) \right] (i,j) \qquad (23)$$

where Adj denotes the binary adjacency matrix associated with G. The equation (23) computes the number of edges in the graph H from the adjacency matrix of G. Note that, in this equation, the power operation is a binary product operation. If G has a high density, the equation (23) shows that $|A_H|$ is almost zero. And implies that for high density graphs, the number of variables and constraints are almost $(nm+n)$ and $(2n+m^2|A|)$ respectively. This result will be confirmed in the next section.

5 Computational results

The numerical results presented in this section illustrates, first the accuracy of the proposed model, second the benefit of pruning the model (the CPU time and the problem size are reduced) and third that our model performs very well on average size problems.

Instances. In order to do the comparisons and prove the benefit of pruning the model, we use two sets of data. The first set, described in Table 1, is used to validate the linear model comparing to the non-linear one and to show the benefit of reducing the size of the linear model. In this table, we have the name and the description of each instance. m is the number of computing units; n represents the number of tasks to schedule and $|A(G)|$ is the number of edges in the graph G. Also the sets have different graph topologies. All the processing time and communication data are generated randomly.

The second group of sets is composed of 1160 instances and are generated randomly with a number of computing units equal to 3 or 5 (2 CPUS and 1 FPGA or 4 CPUS and 1 FPGA), the number of tasks varies from 2 to 250 tasks and with different graph topologies. These sets of data are used to show that our model, with the help of CPLEX, performs very well. They are also

Table 1 First set of data used to compare the models.

Data set	m	n	$\|A(G)\|$
Dataset 1	4	5	4
Dataset 2	4	20	29
Dataset 3	4	20	22
Dataset 4	4	20	24
Dataset 5	5	49	67
Dataset 6	5	49	67

used to show the effect of pruning on the running time and to prove empirically that the size of the model, the number of variables, is almost $O(n)$ for a fixed number of computing units.

In the second group of data, the network is describe by two matrices $(m \times m)$, as in the middle part of the Figure 1, they contain the values of the communication rate and fixed cost. The instances are defined by three matrices: (1) The first one is a $n \times m$ matrix for the processing time, as in the upper part of the Figure 1. It gives for each task the processing time if executed on a given processing unit. If the value is null this means that the task could not be scheduled on the concerned unit. This characterizes to generate the sets F_k. (2) The second one is the adjacency matrix, a $n \times n$ matrix. (3) The third one is, also, a $n \times n$ matrix, it contains the amount of exchanged data between each pair of tasks. The network description used is the same for all instances. The task graph is generated randomly: For each number of tasks (from 2 to 50, 100, 150, 200 and 250), we generate 10 different graphs with different topologies (different densities, average outgoing degrees and average ingoing degrees) and generate the processing time and the exchanged data. The average values, used in the generation process, are based on some observation in our simulation project.

The entire tests are conducted on a laptop with 8 GB of RAM and an Intel processor i7-3740QM with 8 cores. The operating system is a 64-bit Windows 7 professional. For solving the mathematical models, we use CPLEX 12.5x and OPL scripting language.

General results. These experiments aim to validate the models and show their performance. They are conducted on the first set of data.

Table 2 shows the results from comparing the non-linear model with the linear one. The CPU time limit was set to 600 seconds. The C_{max} column is the makespan; the $Time$ column shows the CPU time in seconds and the *gap* column is the gap, in percentage, computed by CPLEX between the best solution and the linear relaxation of the problem. This table shows that for larger problems the non-linear model is unable to find an optimal solution after 600 seconds meanwhile the linear one solves the instance in few seconds.

Table 3 compares the linear model and the reduced linear model in terms of C_{max}, $Time$ and *gap*. In addition, it includes the *nbvar* that represents the number of variables in the model and the *nbconst* that shows the number of

Table 2 The results for comparing the non-linear and the linear models (time limit for the solver: 600 sec).

	Linear Model			Non-linear Model		
Data set	C_{max}	$Time(sec)$	$Gap(\%)$	C_{max}	$Time(sec)$	$Gap(\%)$
Dataset 1	35	0.561	0.00	35	0.265	0.00
Dataset 2	97.25	0.998	0.00	97.25	600.401	2.59
Dataset 3	76.00	5.819	0.00	91.56	619.137	24.12
Dataset 4	49.00	5.897	0.00	64.07	603.069	33.57

constraints in the model. These results substantiate the advantage of pruning the model. The size of the problem is cut by half in almost all cases. The solving time is reduced and even in the case of Dataset 6, CPLEX was unable to solve the linear model within 3000 seconds, but the reduced one is solved in 30 seconds. To emphasize the effect of the pruning on the model, we run a comparative test between the linear model and the reduced one on some instances from the second set. The Table 4 summarizes this effect. In this table, we could notice that the size of the problem is cutted even by 4 for big instances and that the solving time is 29 times shorter for the reduced model in average. The speed-up factor increases for big instances and for graphs with high density.

Table 3 The results for comparing the linear and the reduced models (time limit for the solver: 3000 sec).

	Linear Model				
Data set	C_{max}	$Time(sec)$	$Gap(\%)$	$nbvar$	$nbconst$
Dataset 1	35	0.28	0.00	47	234
Dataset 2	97.25	1.00	0.00	482	3544
Dataset 3	76	5.82	0.00	482	3432
Dataset 4	49	5.90	0.00	482	3464
Dataset 5	152	3000.55	25.62	2648	25293
Dataset 6	191.5	3000.15	0.78	2648	25346
	Reduced Model				
Data set	C_{max}	$Time(sec)$	$Gap(\%)$	$nbvar$	$nbconst$
Dataset 1	35	0.23	0.00	32	111
Dataset 2	97.25	0.53	0.00	239	1589
Dataset 3	76	2.32	0.00	248	1544
Dataset 4	49	2.45	0.00	232	1446
Dataset 5	145	3000.41	15.63	1138	10145
Dataset 6	190	30.61	0.00	1051	6299

Table 4 Effect of the reduction on the solving time and the size of the problem (for the case with 2 CPUs, 1 FPGA and a time limit for the solver of 1200 sec).

	Solving time (sec)			Number of variables		
Nb Tasks	Linear model	Reduced model	Speed-up factor	Linear model	Reduced model	Reduction factor
2-10	0.41	0.28	1	49.95	25.41	2
11-20	1.12	0.54	2	250.84	78.61	3
21-30	42.72	0.70	61	734.67	163.48	4
Average	14.63	0.51	29	346.82	89.75	4

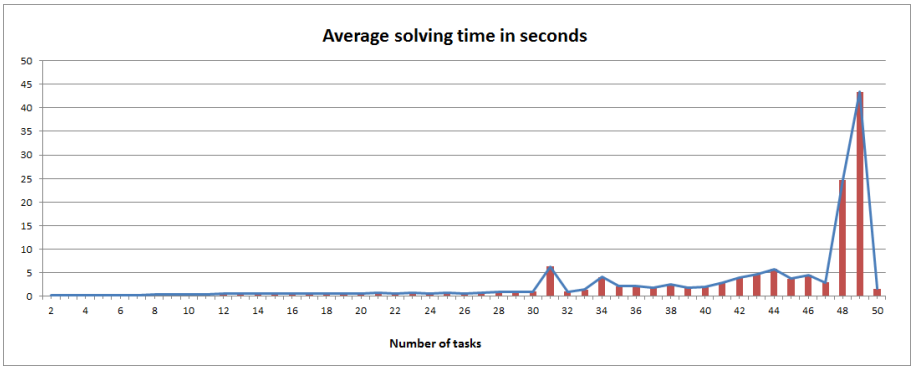

Fig. 2 Solving time for instances with 3 computing units and number of tasks from 2 to 50

Results on average size problems. These experiments use the second set of data for the cases with number of tasks going from 2 to 50 and the number of computing unit is either equal to 3 or 5. These tests were carried out on 1078 cases. Table 5 summarizes the results about the solution time and the number of solved instances within a limit of 1200 seconds. The time is the average value of all the solved instances and it is presented in seconds and in deterministic unit – ticks – used by CPLEX. Figures 2 and 3 plot the solution time in seconds with respect to the number of tasks, respectively for 3 and 5 processing units. We notice, that we were able to solve 98% of all instances and in almost all the cases, the solving time is about few seconds. This proves that our model performs efficiently.

Table 5 Performance on average size instances with tasks from 2 to 50 - 1078 instances (time limit for the solver: 1200 sec)

			Average time on solved instances	
Nb CPUs	Nb FPGAs	Nb solved / total	Time(sec)	Deterministic (ticks)
2	1	537 / 539	2.822	1507
4	1	515 / 539	26.786	13746

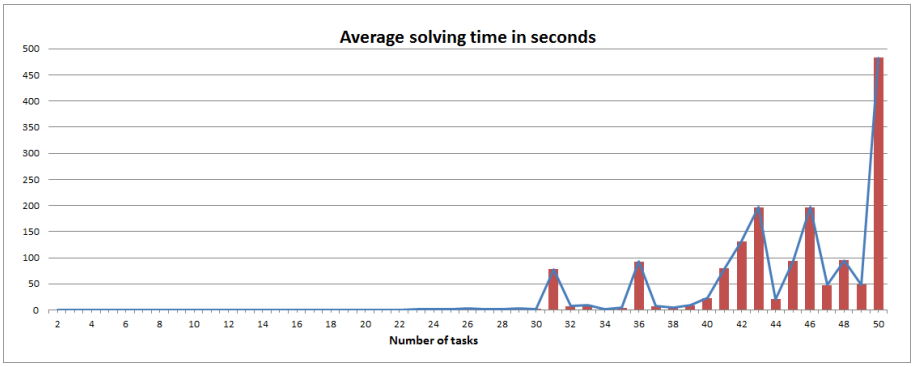

Fig. 3 Solving time for instances with 5 computing units and number of tasks from 2 to 50

Results on large size problems. These experiments use the second set of data for the cases with number of tasks going from 100 to 250 and the number of computing unit is either equal to 3 or 5. These tests were conducted on 82 cases. Table 6 presents for these instances the number of solved instances until the optimum and the average solution time for the solved one. Also, this table provides the average gap for the non solved cases. The results are presented depending on the density of the graph (number of edges divided by $(n(n-1)/2)$, the maximum number of edges). For all these instances, the solver was able to find a feasible solution but a small number is solved until the optimum. For the other instances the gap is interesting especially for the cases with high density.

Table 6 Performance on large size instances with tasks from 100 to 250 - 82 instances (time limit for the solver: 1200 sec)

	Average time on solved instances			Gap on unsolved instances	
Density	Nb	Time(sec)	Deterministic (ticks)	Nb	Average gap (%)
High($\geq 50\%$)	9	272.66	108400	31	15.4%
Low($< 50\%$)	2	105.37	53309	40	25.4%

Empirical estimation on number of variables. These experiments use the entire second set of data composed of 1160 instances.

To emphasize the benefit of the reduction procedure and prove that the number of variables in the model is practically $O(n)$ - for a fixed number of computing units. We run the reduced model on all instances of the second set and gather the information about the number of variables in the model. Figure 4 shows the variation of the numbers of variable with respect to n for graph with density higher than 50%. In addition to the plot, we add the trend line (regression) that fit the most to our curve. The experimental results are of good quality, the coefficient of determination R^2 is almost equal to 1, which prove the validity of the fitting curve. Thus, Figure 4 shows that the number of variables is $O(n)$ empirically for a given number of processing units.

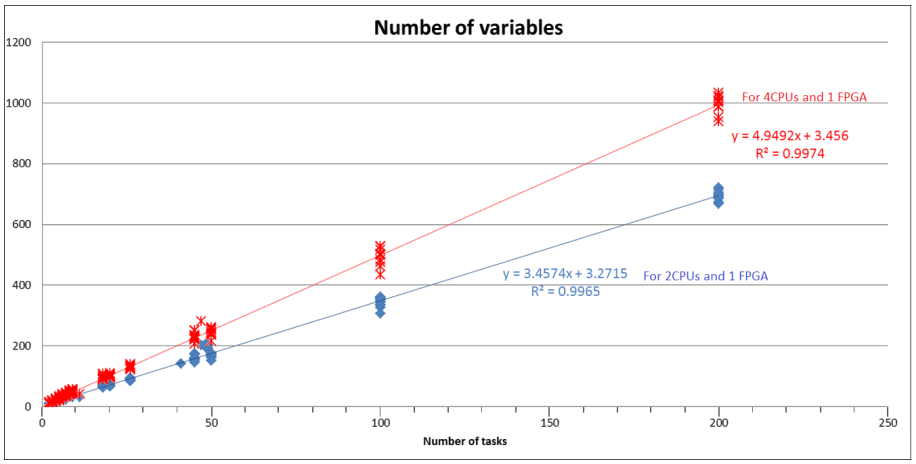

Fig. 4 Variations of the number of variables vs. number of tasks for graph with density higher than 50%.

6 Conclusions

In this paper we propose mathematical models for a scheduling problem in CPU/FPGA architecture with heterogeneous Communication delays. First, we presents a model that takes into account the Communication delays in this heterogeneous environment, i.e. the communication depends at the same time on the tasks exchanging data and the computing units. The resulting model is quadratic due to the communication and the disjunctive constraints. Second, we linearize the model. The linearization of the communication constraints is performed without any additional variables. It was achieved only by using the particularities of our problem. Third, we reduce the linear model by exploiting the precedence graph. The reduction proposed in that model reduces drastically the size of the model. The number of variables is almost $O(n)$ for graphs with higher density if we consider a fixed number of processing unit. The proposed model is promising and could handle average problem with size up to 50 tasks and 5 CPU/FPGA units in few seconds. In our case, we use this model to minimize the C_{max}, but it could be easily adapted to other objectives.

References

1. Huong, G.N.T., Na, Y. and Kim, S.W., *Applying Frame Layout to Hardware Design in FPGA for Seamless Support of Cross Calls in CPU-FPGA Coupling Architecture*, Microprocessors and Microsystems, 35:462-472, 2011.
2. Jong-Kook, K., Shivle, S., Siegel, H.J., Maciejewski, A.A., Braun, T.D., Schneider, M., Tideman, S., Chitta, R., Dilmaghani, R.B., Rohit, J., Aditya, K., Ashish, S., Siddhartha, S., Vangari, P., and Yellampalli, S.S., *Dynamically Mapping Tasks with Priorities and Multiple Deadlines in a Heterogeneous Environment*, Journal of Parallel and Distributed Computing, 67:154169, 2007.

3. Ait El Cadi, A., *Automatisation de la parallélisation de systèmes complexes avec application l'environnement Matlab/Simulink.*, Thèse (M.Sc.A.)-École Polytechnique de Montréal, 2004.
4. Flynn, M., *Some computer organizations and their effectiveness.* Computers, IEEE Transactions on 100(9):948-960, 1972.
5. El-Rewini H., Ali H.H., and Lewis T.G., *Task Scheduling in Parallel and Distributed Systems*, Englewood Cliffs, NJ, Prentice-Hall, 1994.
6. Murty K.G., *Operations Research: Deterministic Optimization Models*, Englewood Cliffs, NJ, Prentice-Hall, 1994.
7. Darte, A., Robert, Y. and Vivien, F., *Scheduling and Automatic Parallelization*, Birkhäuser BOSTON 2000.
8. Baker, K. R. and Trietsch, D., *Principles of Sequencing and Scheduling*, Wiley, 2009.
9. Hartmann, S. and Briskorn, D., *A Survey of Variants and Extensions of the Resource-Constrained Project Scheduling Problem*, European Journal of Operational Research, 207:1-14, 2010.
10. Banharnsakun, A., Sirinaovakul, B. and Achalakul, T., *Job Shop Scheduling with the Best-so-far ABC*, Engineering Applications of Artificial Intelligence, Volume 25(3):583593, 2012.
11. Chen, W.H. and Lin, C.S., *A Hybrid Heuristic to Solve a Task Allocation Problem*, Computers And Operations Research, 27(3):287-303, 2000.
12. Long, Q., Lin, J. and Sun, Z., *Agent Scheduling Model for Adaptive Dynamic Load Balancing in Agent-based Distributed Simulations*, Simulation Modelling Practice and Theory, 19:1021-1034, 2011.
13. Korkhov, V.V., Moscicki, J.T., and Krzhizhanovskaya, V.V., *Dynamic Workload Balancing of Parallel Applications with User-level Scheduling on the Grid*, Future Generation Computer Systems, 25:28-34, 2009.
14. Garey, M.R. and Johnson, D.S., *Computers And Intractability: A Guide to the Theory of NP-Completeness*, WH Freeman & Co., San Fran-cisco, 1979.
15. Koné, O., Artigues, C., Lopez, P. and Mongeau, M., *Event-based MILP Models for Resource-Constrained Project Scheduling Problems*, Computers & Operations Research, 38:3-13, 2011.
16. Urban, T.L., *Note. Optimal Balancing of U-Shaped Assembly Lines*, Management Science, 44(5):738-741, 1998.
17. Catalyurek, U.V., Boman, E.G., Devine, K.D., Bozda, D., Heaphy, R.T. and Riesen, L. A., *A Repartitioning Hypergraph Model for Dynamic Load Balancing*, Journal of Parallel and Distributed Computing, 69:711-724, 2009.
18. Gen, M., Lin, L. (2014). *Multiobjective evolutionary algorithm for manufacturing scheduling problems: State-of-the-art survey.* Journal of Intelligent Manufacturing, 25(5), 849866.
19. Ali H., H., El-Rewini, H., *An optimal algorithm for scheduling interval ordered tasks with communication on N processor*, University of Nebraska at Omaha, Math. And Computer Science Department, Technical Report, 9120, 1990.
20. Luo, H., Zhang, A., Huang, G. Q. (2013). *Active scheduling for hybrid flowshop with family setup time and inconsistent family formation.* Journal of Intelligent Manufacturing, 1-19.
21. Vázquez, E. P., Calvo, M. P., Ordóñez, P. M. (2013). *Learning process on priority rules to solve the RCMPSP.* Journal of Intelligent Manufacturing, 1-16.
22. Hao, X., Gen, M., Lin, L., Suer, G. A. (2015). *Effective multiobjective EDA for bi-criteria stochastic job-shop scheduling problem.* Journal of Intelligent Manufacturing, 1-13.
23. Cakici, E., Mason, S. J., *Parallel machine scheduling subject to auxiliary resource constraints*, Production Planning and Control, 18:217225,2007.
24. Zhang, W., Xu, W., Liu, G., Gen, M. (2015). *An effective hybrid evolutionary algorithm for stochastic multiobjective assembly line balancing problem.* Journal of Intelligent Manufacturing, 1-8.
25. Chrétienne, P., Picouleau, C., *Scheduling with communication delays: A survey*, In: Chrétienne, P., Coffman, E.G., Lenstra, J.K., Liu, Z. (eds.) Scheduling theory and its applications, pp. 6590. John Wiley & Sons, New York, 1995.

26. Zhang, W., Gen, M., Jo, J.B. (2014). *Hybrid sampling strategy-based multiobjective evolutionary algorithm for process planning and scheduling problem.* Journal of Intelligent Manufacturing, 25(5), 881897
27. Zhang, S., Wong, T. N. (2014). *Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning.* Journal of Intelligent Manufacturing, 1-17.
28. Dauzère-Pérès, S., Sevaux, M., *Using Lagrangean relaxation to minimize the weighted number of late jobs on a single machine*, Naval Research Logistics, 50(3):273288, 2003.
29. Davidović, T., Hansen, P., Mladenović, N., *Permutation-based Genetic, Tabu and Variable Neigh-borhood Search Heuristics for Multiprocessor Scheduling with Communication Delays*, Asia-Pacific Journal of Operational Research, 22(3):297326, 2005.
30. Davidović, T., Liberti, L., Maculan, N., Mladenović, N., *Towards the optimal solution of the multiprocessor scheduling problem with communication delays*, MISTA Proceedings, 2007.
31. Harris, J.M., *Combinatorics and Graph Theory*, New York, Springer-Verlag, 2000.
32. Hwang, R., Gen, M., Katayama, H., *A comparison of multiprocessor task scheduling algorithms with communication costs*, Computers & Operations Research, 35:976993, 2008.
33. Isaak G., *Scheduling rooted forests with communication delays*, Order 11:309316, 1994.
34. Knuth, D.E., *The Art Of Computer Programming* Vol 1. 3rd ed. Addison-Wesley, Boston, 1997.
35. Luo, P., L, K., Shi, Z., *A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems*, Journal of Parallel and Distributed Computing, 67:695714, 2007.
36. Pinedo, M., *Scheduling: Theory, algorithms, and systems*, 2nd ed. Prentice-Hall, New Jersey, 2002.
37. Prastein, M., *Precedence-constrained scheduling with minimum time and communication*, MS Thesis, University of Illinois at Urbana-Champaign,1987.
38. Rayward-Smith, V.J., *UET scheduling with unit interprocessor communication delays*, Discrete Applied Mathematics, 18:557, 1987.
39. Sousa, J. P., Wolsey, L. A., *A time-indexed formulation of nonpreemptive single machine scheduling problems*, Mathematical programming, 54:353367, 1992.
40. Unlu, Y., Mason, S. J., *Evaluation of mixed integer programming formulations for nonpreemptive parallel machine scheduling problems*, Comput. Ind. Eng., Pergamon Press, Inc., 58:785800, 2010.
41. Venugopalan, S., Sinnen, O., *Optimal Linear Programming Solutions for Multiprocessor Scheduling with Communication Delays*, In: Xiang, Y., Stojmenovic, I., Apduhan, B.O., Wang, G., Nakano, K., Zomaya, A. (Eds.) Algorithms and Architectures for Parallel Processing, Springer, Heidelberg, 7439, 129138, 2012.