On line HW/SW Partitioning and scheduling for data dependent execution time applications

GHAFFARI Fakhreddine *& ; BENJEMAA Maher**; ABID Mohamed**; AUGUIN Michel***
*I3S, University of Nice Sophia Antipolis, CNRS, *les Algorithmes bat. Euclide, 2000, route des Lucioles BP 121, 06903 Sophia-Antipolis Cedex*
***Research unit GMS. Ecole Nationale d'Ingénieurs de Sfax, BPW 3038 Sfax Tunisie.*
{ ghaffari, auguin}@i3s.unice.fr ; {maher.benjemaa, mohamed.abid}@enis.rnu.tn

1. INTRODUCTION

Functional partitioning is becoming an increasingly important task in the system design flow. In functional partitioning, the functions of the behavioral specification are assigned to system components, which include standard software processors, custom hardware processors, and memories.
Many applications, in particular in image processing (e.g. an intelligent embedded camera), have data dependent execution times according to the nature of the input to be processed.
This kind of applications is often stressed by real time constraints that impose adapted computation capabilities. To partition data-dependent tasks on a heterogeneous architecture, new design approaches are necessary. Particularly for application with soft real time constraints we aim to minimize the embedded resources so as to avoid an architecture based on the composition of the resources associated with the worst case execution times of the functionality of the application. There is little work in the literature which addresses this problem. The approach presented in [1] is based on an on-line HW/SW migration of tasks according to their execution times. This migration process is only applied locally to the most time consuming loop of the application program.
The choice of dynamic re-allocation of the tasks presented in [2] is manual.
We present here a new approach of partitioning/scheduling for time-dependent tasks.

2. Dynamic Hw/Sw reconfigurable platform

We focus on soft real-time application that operates iteratively on a flow of data (e.g. image processing). The application is described by a data flow graph (DFG) where the nodes represent the functions or tasks of the application. Figure 1 depicts our embedded on line dynamically reconfigurable platform. The platform is composed of i), an execution unit including a processor connected to a dynamically reconfigurable hardware unit ii), a data base used to estimate execution time of tasks [3] iii), a prediction unit that evaluates the total execution time of the whole application and iv), a partitioning/scheduling unit

that provides a valid mapping and schedule of the tasks according to the time constraint Tmax.

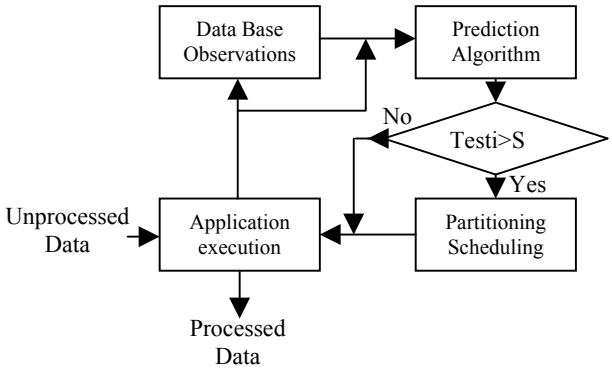

Fig 1: HW/SW dynamic reconfigurable platform

For each task a Sw and/or a Hw implementations are stored in the platform (program code and bit stream).
These implementations are built off line.
The normal execution flow of the platform on a sequence of input data I_{n-1}, I_n, I_{n+1},... is illustrated in figure 2. During period I_{n-1} the execution times of each tasks and of the whole application (Te) for the next period are estimated. While it is estimated that no violation of the time constraints will occur ($T_e < T_{max}$), the current mapping and scheduling hold for the next period. Otherwise, a new partitioning/scheduling of the tasks is computed as depicted in period In in figure 2. The new mapping must deal with the available resources in the reconfigurable hardware unit and must provide a solution with an execution time less than the time constraint.

The great challenge in this approach is to develop efficient and rapid algorithms for execution time prediction, partitioning and scheduling, The on line partitioning approach is described in the next

section and the on line scheduling algorithm is presented in section 4.

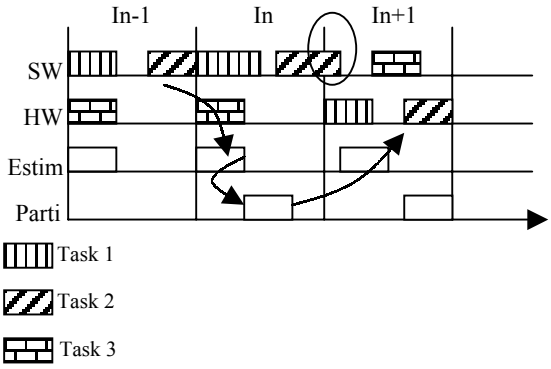

Fig 2: Adaptation of partitioning to the processing

3. Dynamic partitioning according to processing requirements

Partitioning occurs when violation of constraints are predicted. The basic idea consists to consider the current mapping and to carry out migrations between implementations to satisfy the overall constraints (see period In+1 in figure 2).

Assume that a time constraint violation is predicted at period I_n and that the task T_i is assigned to the processor. The current available area in the HW unit is S_n. A migration of T_i to the HW at the next period leads to a benefit of Δ_i on its execution time and the available area will be reduced to $S_{n+1} < S_n$. For each task T_i the values $Ci = \Delta_i \ .S_{n+1}$ are evaluated. Potential migrations are analyzed in the decreasing order of C_i. The aim is to move to the Hw the task which provides the largest acceleration but that minimizes the reduction of the available Hw resources. For each potential migration a schedule is computed (see section 4). The migration is rejected if only a limited benefit on the global execution time is expected. In this case the next potential migration is considered and this partitioning scheme proceeds until the time constraints are met. If no migration is enabled due to a too weak number of available resources in the HW, the partitioning process starts with a migration to the SW of the task mapped on HW so that the benefit in resources is maximized and the time penalty is reduced.

Each time a migration is considered, it is necessary to evaluate its time benefit on the global execution time. This evaluation is performing with an on line a scheduling of the tasks.

4. On line scheduling technique

The problem of task scheduling on several processors is a NP-hard optimization problem. The on line scheduling can be addressed only with a fast heuristic. The problem is to get efficient solutions with a low complexity algorithm.

The basic idea consists in scheduling the tasks of the graph according to three criteria: ASAP (As Soon As Possible), urgency and execution time. A task is urgent when at least one of its successors is mapped on HW. A schedule is constructed by traversing the DFG from the roots. From the tasks ready to be executed we schedule the task with the lowest ASAP time. If more than one task have this same ASAP time, the urgency is considered next. The task which has the highest urgency is executed first.

If two tasks have both the same ASAP and the same urgency, the task with the highest execution time is scheduled first..

When all the tasks in the DFG are scheduled, we get an evaluation of the execution time of the whole application. This time is either compared with the time constraints in the prediction algorithm or considered in the partitioning heuristic to accept or reject a potential migration.

5. Conclusion

Compared with a traditional off line partitioning approach, dynamic hardware/software partitioning can do a more efficient use of the available resources in the system since it deals with the actual execution times rather than with the worst case execution times. However, on line partitioning imposes additional resources to implement the partitioning and scheduling heuristics. Our first objective is to compare this overhead with the benefit on executive resources induced by the on line partitioning approach. The behavior of the complete system on a real application will be studied in simulation with SystemC. The objective is to evaluate the ability of the partitioning and scheduling heuristics to adapt the mapping of the tasks according to slow or fast variations of the execution times.

6. References

[1] Roman Lydecky, Frank Vahid “A configurable Logic Architecture for Dynamic Hardware/Software Partitioning”. In Proc. of the DATE 2004 Conference. Paris, February 2004.

[2] J-Y. Mignolet, V. Nollet, P.Coene, D.Verkest, S.Vernalde, R. Lauwereins “Infrastructure for Design and management of Relocatable Tasks in a Heterogeneous Reconfigurable System-on-chip”. In Proc. of the DATE 2003 Conference. Messe Munich, Germany March 3-7, 2003

[3] F.Ghaffari, M.Abid, M.Auguin M.Benjemaa. “Algorithme de prédictions statistiques du temps d’exécution basée sur la méthode de KPPV”, SCS’2004, pp. 265-271 Monastir, Tunisie 18-21 Mars 2004.