An Adaptive On-line HW/SW Partitioning for Soft Real Time Reconfigurable Systems

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

Abstract

We present here a new HW/SW partitioning approach. This partitioning method is called an On-line Partitioning Algorithm (O.P.A) which consists to adapt dynamically the architecture to the processing requirements. A scheduling heuristic is associated to this partitioning approach. We compare our method with an Off-line static HW/SW partitioning approach.

Index Terms— HW/SW Partitioning, Dynamic Scheduling, real time constraints, ILP formulation.

1. Introduction

Instead of creating more powerful embedded processor to meet the ever increasing computational demands of multimedia applications, we could consider adding reconfigurable logic to create flexible accelerators. In this way heterogeneous applications could be constructing using tasks that are capable of executing either on the embedded processor, on the reconfigurable logic or on both. The aim is to increase the computation power of current multimedia portable devices (such as embedded camera) while keeping their flexibility. Hardware/Software partitioning is the process of dividing an application among software (running on a microprocessor) and hardware co-processors. Extensive research has shown that Hardware/ Software partitioning can result in overall software speedups [5] as well as reducing system energy [6].

Many applications, in particular in image processing (e.g. an intelligent embedded camera), have dependent data 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 a maximized architecture based on the worst case execution times (WCET) of the tasks. 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. In [3] an approach of tasks re-allocation between hardware and software units has been presented. It details the principle of the communication after switching from implementation to another.

The primary contribution of our work, though, is an extensive examination of the numbers of hardware resources savings as well as speedups possible through hardware/software partitioning. We have simulated our approach with System C on processing image application. We have analyzed speedups savings by using estimation models, as well as by taking physical measurements of real platforms. We have examined partitioning for a single-chip platform having a microprocessor coupled with a configurable logic.

The paper is organized as follow. Section 2 presents the on-line partitioning algorithm. Section 3 shows the experimental results and finally we conclude in the section 4.

2.The On-line Partitioning Algorithm: O.P.A

On-line HW/SW Partitioning has several important advantages over off-line approaches. O.P.A allows for a system to be optimized based on runtime behaviors and values, which may be hard to determine using off-line methods or costly simulations. Furthermore, on-line optimizations require no designer intervention and are applied transparently during runtime.

In application such as image processing, the execution of data dependent tasks consumes time according to the

Proceedings of the 2005 8th Euromicro conference on Digital System Design (DSD'05)
0-7695-2433-8/05 $20.00 © 2005 IEEE

characteristic of the input image. The time variation results from one or more Correlation Parameters (C.P) in every processed image. Example of C.P is the number of white pixels in an image or the number of moving objects. The goal of the On-line Partitioning Algorithm (O.P.A) is to dynamically allocate resources of the architecture and schedules the tasks such that time constraint violation are minimized and the set of embedded resources are minimized. The figure 1 illustrates the functionalities embedded in an OPA system composed of two parts: the application itself and the O.P.A loop. The former is data flow oriented such as video, image, sound or any iterative data processing application which have data dependent execution time. The application has its own internal data communications between its tasks, but only the exchanged data with the O.P.A loop subsystems is shown. The tasks of the application are executed according to an order given by the Scheduler, which also collects the application measured times and C.P. Once the application reaches the end of the iteration, the O.P.A lunches the Estimating function: thanks to all the measured data, it will make estimation on execution time of each application task for the next iteration. If the estimated schedule respects the timing constraints, the O.P.A loop will stop and lets the Executing function runs the next iteration. In the other case the algorithm will run the Partitioning function and tries to find a partition which respects the constraints. This global view of the algorithm leads to question about its integration in the whole system, and how complex will it to be implement.

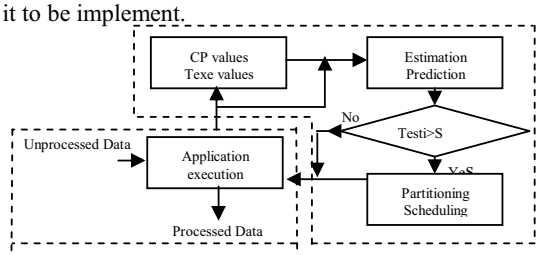

Fig 1: Dynamic reconfigurable platform

The OPA approach need for the implementation an architecture which insures the migration of tasks from the processor to the reconfigurable and from reconfigurable to the processor. For this, the target architecture is composed of a processor connected to a Reconfigurable Computing Unit (RCU) through an intelligent interface ICURE conceived by the CEA.

The considered embedded processor can be anyone type provided that allows an efficient coupling with the RCU. The reconfigurable unit contains a logic cells (LC) which can be reconfigured to realize any task. The RCU reconfiguration is achieved by a dedicated unit situated in the ICURE interface.

a. System Model

To model application we consider a Data Flow Graph (D.F.G), where nodes are the processing steps and edges are communication between tasks. A low granularity D.F.G makes the system easy to be predictable because data dependent task execution time can depend on a single C.P. In opposite a high granularity D.F.G needs more calculations to complete the algorithm loop, and it can exceed the iteration cost constraints. Another difficulty lies on having a good ratio between the application cost and the O.P.A cost. In fact, the algorithm execution costs (time and resources) must be insignificant as compared to the application; else, it will not be benefit to overload the system with the O.P.A. To resume, the Online Partitioning Algorithm tries to update the partition of an application when it is running. The algorithm must make choices on tasks migration.

b. Prediction algorithm

The stability of the system depends on the estimation results. With the estimated times, for the next iteration, the O.P.A decides if the partition has to be changed. To estimate the future execution time of a task with variable execution time (Texe), there are two main approaches: simple interpolation equation and K-nearest neighbors (KNN) [4]. We use the simple interpolation method when the execution time depends only of one correlation parameter; and we used the KNN method to estimate execution time of tasks which depend of more than one CP. The All Implementation Context is called when the Partitioning function must be done.

c. DataBase updating (DB)

The database is used only with the KNN estimator to store the past observations (task's execution time) and the CP values. The Database stores real execution time values and their corresponding C.P of each task according to its implementation. A task has a minimum of one implementation: software or hardware. But it could have more than one implementation and generally for hardware tasks, more are the dedicate hardware resources allocated for a task implementation less will be its execution time.

d. Partitioning Algorithm

The partitioning is required when a violation of the constraints is predicted. The O.P.A realizes a migration of tasks from SW to HW in order to lesser the execution time down to the time constraint limit. When the processing units remain idle at the end of execution of the current iteration for enough time, OPA performs a HW to SW migration in order to free resources in the RCU.

1. Up-speeding migration:

The iteration time constraint includes the application execution time and the O.P.A execution time. The second should be small against the first one. For a video application, iteration takes 40ms; this should

include the image processing and the O.P.A execution. If the O.P.A maximum cost is 1 ms, then the application is allowed to run for 39 ms. So if the application execution is predicted to last more than 39 ms, then the Partitioning must accelerate one or more tasks to reduce that cost. It is interesting to prevent the iteration lost by reducing the application execution high limit: for example 35 ms. Then the O.P.A will try to do a partitioning before the next iteration.

On line partitioning is realized in OPA on a migration process that changes the allocation of tasks, one task at once, until time constraints are met. Indeed, the partitioning consists of an updating of the current mapping in order to take into account local variations of execution times. The migration of a task consists of choosing a faster implementation between the set of contexts embedded in the system. The efficiency of the online partitioning depends on the right speed-up vs resources implementations selected in an offline profiling analysis and synthesis. For each potential migration the algorithm evaluates the parameter G defined as the product of the time benefit and the number of resources remaining free after migration.

We start by sorting the possible implementations by the decreasing order of G. If this up-speeding migration didn't satisfy the time constraint then we added the next candidate migration until the temporal constraint is satisfied or when the reconfigurable is saturated. We then created a version of the application with all the critical tasks moved to the reconfigurable hardware.

2. Up-freeing migration

This kind of migration is necessary to avoid the saturation of the reconfigurable caused by the successive SW→HW migrations. The method is analogous to the first migration. We choose the task which has the minimum benefits of time and which can free the maximum of hardware resources. The partitionner have to repeat this process until that the total execution time to be between the HL (High Level) and the LL (Low Level).

e. Scheduling Algorithm

To evaluate the partitioning solution, we must calculate the total execution time of the application. The problem is when there are several software tasks in the application; consequently the application may have two tasks requiring the processor at the same instance. With the estimated execution times and the application partition, the scheduler algorithm chooses an order to run all the software tasks.

The task can have two states: waiting and scheduled. The waiting state remains until all the task predecessors are scheduled. If a task is to mapped on hardware, then it goes in scheduled state. If it will be run on software, then the scheduler need to check if the timing allows the task to be scheduled. There are three possible cases:

1. when the software resource is free with only one software task to schedule,
2. when the software resource is free and there is more than one software task to schedule,
3. and when the processor is busy.

In the first case, the task is scheduled because it is the only one which requires the processor. In the third case, all the tasks wait for the software resource to be free. In the second case, the scheduler must set up a priority table to choose the software task to be scheduled first. All the others software tasks will become dependant to the chosen one. The choice of the priority is based on the successors of the task. The one which has the most critical successor will be attributed the highest priority. A task is critical if it has at least one hardware successor. The highest estimated execution time of all the hardware successors determines the first level of priority. If there is still more than one task having the same priority then the second level of priority is the one that has the lowest execution time.

View the complexity of the scheduling algorithm, we choose to implement it on hardware. Thus the time computing of the scheduling algorithm is neglected.

The scheduler architecture is fully synthesisable, and it was done with a XILINX Virtex II Pro FPGA tools. The size needed (number of CLB) depends upon the number of tasks, and the complexity of theirs interconnections. The synthesis is done on a Virtex II Pro vp100: Whole Scheduler block takes 1677 slices out of 44096 which represents 3%. The results satisfy the requirements for this O.P.A, which is minimising the cost of the Estimation-Scheduling-Partitioning time consuming.

3. Experimental Results

We have simulated our partitioning approach with SystemC on image processing application. The motion object detection on fixed image background requires resources for data processing in order to process images in real time. ICAM (Intelligent CAMera) is an algorithm developed by the C.E.A[1], used for embedded camera to detect objects motion. Such application can be used for parking supervising, identification, and pieces selection according to the shape…

ICAM is considering as an application with twenty tasks. Each of them could be run on software as well as on hardware. The model in SystemC of this application must have the software and the hardware parts. On the software part, each task is a function which is called by the processor when needed. On the hardware side, each task is an independently hardware block with inputs and outputs for data communications. The application running is controlled by the kernel function from the O.P.A definition, thus the software and hardware parts

[1] CEA : Commissariat à l'Energie Atomique (Research Comitee on atomic energy)

of the ICAM have a direct communication with the Kernel function. The communication with memories is only for image data, the communication with the Kernel function is control data, measured execution times and C.P. The execution time is measured with the simulating platform processor clock: the modeling and simulating have been done under a Pentium Centrino 1.5 GHz with MS Windows XP as O.S. The measured times are achieved by a processors A.S.M (1) instruction which can give time in microsecond precision. Only software execution times are measured.
The following table shows the execution time of all OPA functions:

Algorithm	Average execution time (ms)
Partitioning	0.33003
Scheduling	0.002 (HW implement)
Estimtor	0.42166
DataBase	0.01033
Application	33.5652

According to the table above, we notice that the OPA timing cost is neglected as compared with the execution time application. We remind that the OPA approach has for objective to adapt on-line the partitioning result with the processing requirements. This is showed clearly in the figure 2: the OPA limits always the application execution time between the high level time and the low level time by updating the HW/SW partitioning result.

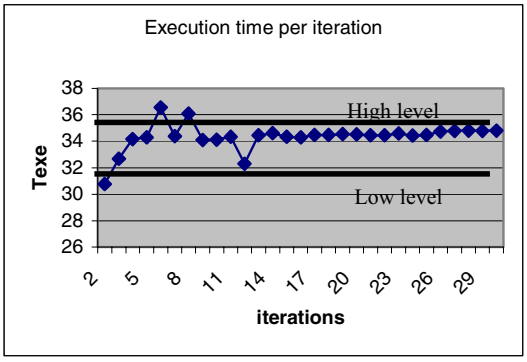

Fig2: Total execution time per iteration

We compare our partitioning approach with an optimal off-line partitioning method based on ILP (Integer Linear Programming) solver. The ILP based approach can solve the partitioning problem optimally. The partitioning problem is formulated as follows:
The objective function is to minimize the number of hardware resources to be used by the application. The main constraint is that the total execution time must respect the real time constraint. The ILP is a static partitioning method based on the Worst Case Execution Times (WCETs) of the application tasks. As it is assumed in OPA, the communication delays between tasks are neglected.

The table hereafter shows the differences between the two approaches results:

Approa	Resources used	Nbre Image lost	Off-line work	Resource added	flexibl
ILP	852	0	Yes	No	No
OPA	300	0.35%	limited	Yes	Yes

As depicted on the table above, ILP demands more resources for the application than OPA since it tries to reach the optimal solution. The OPA approach tries only to satisfy the constraints.

4. Conclusion

Results of our approach show the efficiency of the adaptation of partitioning to needs of treatment. The dynamics reconfiguration of the FPGA allows the architecture to accept several contexts of reconfiguration as results of HW/SW partitioning.
A dynamic HW/SW partitioning approach has many advantages over traditional partitioning approaches. Dynamic partitioning can adapt to an application's constraints dynamically at run time. We presented a HW/SW partitioning approach based on on-line reallocation tasks. We also presented our approach of scheduling based on a criticality parameter chosen on-line. Our future work consists in validating these approaches on real applications by simulation then by execution on an industrial platform.

5. 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] Stitt, G., Lysecky, R., Vahid, F. Dyanmic hardware/software partitioning: a first approach. Proceedings of the 40th ACM/IEEE Conference on Design Automation (DAC), 2003.

[3] 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

[4] F F.GHAFFARI, M.BENJEMAA, M.AUGUIN. Algorithms for the Partitioning of Applications containing variable duration tasks on reconfigurable architectures. IEEE Int. Conf. AICCSA 2003 Tunis, TUNISIA 14 – 18 July 2003.

[5] Balboni, A., W. Fornaciari and D. Sciuto. Partitioning and Exploration in the TOSCA Co-Design Flow. International Workshop on Hardware/Software Codesign, pp. 62-69, 1996.

[6] Henkel, J., Y. Li. Energy-conscious HW/SW-partitioning of embedded systems: A Case Study on an MPEG-2 Encoder. Intl. Workshop on HW/SW codesign, 1998.