A Grouping Partitioning Technique with Automatic Criterion Selection
for the Codesign Process.

J.A. Maestro, D. Mozos, J. Septién
Dpto. de Arquitectura de Computadores y Automática
Universidad Complutense - 28040 Madrid, Spain
http://www-ac.dia.ucm.es/~jam

Abstract

Partitioning is a key issue within the Codesign process. As it usually consumes much time, it is important to make it as efficient as possible to avoid costly feedbacks. In this way, this paper describes a partitioning process based on a grouping technique intended to avoid high time overheads due to communications, as well as being able to work with a reliable estimation model.

1.- Introduction.

In the recent years, and due to the embedded system market expansion which demands more exigent requirements, the Codesign process has become a major trend in many research fields. In this way, a whole system design cycle is intended, rather than a hardware design, as the High Level Synthesis pursued.

Many advances have been achieved in this area, due to a high research activity. However, there are still important issues to improve, especially when facing designs with a hard set of constraints.

One of the key problems widely referenced is partitioning [1]. Although there exists a lot of well-proved algorithms that carry out this task, there are some reasons that make necessary to revise them:

• The existing algorithms usually work with classic estimation models. However, there are more factors to take into account [2], as multiple hardware implementations for a single node, task parallelism and hardware sharing. In this way, the classical node moves between the hardware and the software partitions are now extended to meet these new features.

• Most of the algorithms have been developed in very different fields from Codesign. Thus, they lack of the necessary mechanisms to avoid the typical problems of this area. The time overhead due to high communications and the presence of hard constraints can produce a deep distortion in the design space.

As far as we know, there is not any system that benefits from the previous features. Taking all of them into account, we present a partitioning technique able to deal with these considerations in a reliable way.

2.- Related work.

There are many researches on partitioning for a Codesign environment.

Kalavade and Lee [3] propose a new partitioning technique, in which two different objective functions are tried to minimize, selecting one of both depending on the proximity of the current design to the proposed goal. To achieve this, and considering their inner characteristics, the system functionalities are divided into three different types. However, nothing is said about the estimation process of their inner time and cost parameters.

Niemann and Marwedel [4] use an Integer Programming algorithm to perform the partitioning task. Although this approach offers an exact solution, the long time required makes it unsuitable for complex problems. Talking about the estimations, these are made by a software compiler and a High Level Synthesis tool which have to carry out most of the final design to obtain the parameters.

Eles et al. [5] start from a net of VHDL processes, and then, two stages are performed to fulfill the partitioning task. First, the set of processes that are candidates to be mapped into hardware are decided. This is done by calculating the relative computational load of each one with respect to the whole program. Second, and beginning from the previous result, a standard partitioning algorithm is executed to get at the final solution. At this moment, the algorithms of Simulated Annealing and Tabu Search are proposed.

This work has been granted by CICYT TIC 96/1071 and HCM CHRX-CT94-0459
3.- Problem definition.

According to our approach, the Codesign partitioning problem can be stated as follows:

Given a Codesign graph $G=\{N,E,K,P\}$ and a time constraint, $t_s$, find the mapping function $S$ and the set of forced edges, $E_f$, so that the overall time meets the imposed constraint with a minimum system cost.

- $N$ stands for the set of nodes.
- $E$ is the set of graph edges.
- $K$ is the sharing matrix $K_{[i,j][2,2]}$, in which for every pair of node and hardware implementation, $[i,j]$, its similarity with the rest of the pairs is stated.
- $P$ is a superset of node estimations, in which for every node, the parameters of time and cost related to the software implementation and to some of the existing hardware implementations are stated.
- $S$ is a function which assigns each node $i$ to the software partition or to one of the different $m(i)$ possible hardware implementations.
- $E_f$ represents the set of forced hardware links.

For a detailed explanation of these elements, refer to [2]. With the previous considerations, the graph can represent our time and cost estimation process, which allows task parallelism and hardware sharing, among other features.

4.- About the partitioning process.

Most of the partitioning algorithms can be classified as iterative. They usually try to find at any time the most suitable movement by comparing the partial state with the previous ones. One of the most used algorithms is the Fiduccia-Mattheyses (FM) [6], that is based on the min-cut process.

An important problem that can arise in Codesign typical systems is the communication between the two existing partitions. If we are dealing with a design that has an intensive interchange of parameters, the time dedicated to this labor can be a high portion of the overall execution time. This means that if two nodes with a high data interchange are assigned to different partitions, the associated communication will be triggered off, creating a great time overhead and making the groups that contribute to reduce the algorithm efficiency, or in other words, to move the nodes related to this problem at the same time. This feature of simultaneous movements implies the modification of the iterative algorithms, that are based, as it was said before, on single-node moves.

However, attempting this could have a negative impact on the efficiency of the process, since modifying its code would probably lead to a worsening in its performance. Thus, it would be desirable to adapt the algorithm to the proposed characteristic, without altering its well-known efficiency. The only possibility to do that is by modifying the input node space according to our convenience.

Therefore, the original nodes of the graph would be grouped into higher entities, following a predefined criterion. These groups would be moved as a whole by the partitioner, and the inner communications would never be triggered off, eliminating their negative effect.

This process could be considered a classic-approach clustering, since it reduces the number of nodes, as well as the problem complexity. However, it has certain differences that change its intrinsic meaning:

- Its importance does not lie in decreasing the complexity of the problem, but in avoiding undesirable effects on the iterative partitioning. In this way, it is not a kind of partitioning previous to the main algorithm.
- The formed groups are abstract entities without inner structure, with the only characteristic of being moved simultaneously through the partitioning steps. In this way, the groups are not clusters but linking structures.
- Once the problem has been stated together with our proposed approach, the variation of the algorithm will be explained more thoroughly.

5.- Extension of the FM algorithm.

In this section, the new steps originated by our approach are compared with the classic FM algorithm.

The core of this algorithm resides in an inner loop, which tries to find the best possible movement among the available ones. These are reduced to the possibility of migrating a node from hardware to software or vice versa. But in our case, a movement is mathematically expanded to the 4-tuple:

$$\mu = (i, e, Pred_f_i, Suc_f_i)$$

where $i$ stands for the node that is going to be moved.

$e$ is the implementation of $i$; $e \in \mathbb{Z}$, which was defined in [2] as the set of possible implementations for node $i$.

$Pred_f_i$ is the set of forced edges from $i$'s predecessor to $i$, and $Suc_f_i$ is the set of forced edges from $i$ to $i$'s successors. In this way, $Pred_f_i \cup Suc_f_i = E_f \cap [i]$, where the last term represents all the forced edges in which $i$ takes part.

Notice that this definition expands the classic concept of movement, in which the only important fact was choosing the node to move, since the only possibility was swapping from hardware to software or vice versa.
By the previous concepts, there are three kinds of possible movements:

a) SW → HW: From software to one of the possible hardware implementations.

b) HW → SW: From hardware to software.

c) HW_i → HW_j: From a hardware implementation to a different one.

The last case was not considered as a movement in a traditional approach. However, it is essential to make the partitioner evolve toward the best solution. These movements are applied on the clusters obtained after the grouping process.

6.- Grouping process.

This step is in charge of determining the grouping criterion, $\tau$. The main factor to consider is the communication load between the pairs of nodes. The problem here resides in determining which of these communications will create problems to the partitioner, or in other words, the threshold to consider a communication as high.

There are many factors that influence the grouping criterion, aside from the intrinsic overhead of every communication, $o(c)$. In this work, we have determined $\tau$ by taking some macroscopic factors into account:

- The time constraint $t_r$. As we always start with an all-software state, the harder $t_r$ is, the more nodes will be migrated to hardware, triggering their communications in this process.

- The probability of triggering a certain communication, that is, the probability, $\pi$, of moving its adjacent nodes. If a communication is going to remain inactive, it does not matter whether it is grouped or not.

- The potential number of active communications in the system. The more number of them, the more influence on the overall system time. To soften this, the correction factor, $\gamma$, is applied.

The probability $\pi(c)$, for a given communication $c$, is defined in this way, being $i,j$ the adjacent nodes:

$$\pi(c) = \frac{\pi(i) + \pi(j)}{2}$$

$\pi(i)$ represents the probability of moving the node $i$, under the conditions determined by $t_r$. It is a relation between the possible time and cost gains by moving that particular node to hardware. If the constraint is very hard, a maximum time gain is attempted, without regarding the possible cost repercussion, and vice versa, if the constraint is nearly or already met.

$$\pi(i) = \frac{f_r(i) \cdot r_r(i) - f_r(i) \cdot r_c(i)}{T_{max} - C_{max}}$$

where $r_r(i)$, $r_c(i)$ are the average time and cost gains obtained if $i$ is moved from software to any of the multiple hardware implementations; $T_{max}$ and $C_{max}$ are the maximum time (all the nodes in software) and maximum cost (all the nodes in hardware) of the system; $f_r$ and $f_c$ are real functions in the range $[0,1]$ that change depending on the partitioning stage, and drive $\pi$ to a time or cost reduction policy. They measure the distance from the actual system time to the imposed time constraint. The function $f_r$ is depicted in Figure 1, where there are some factors to model its curvature; in the same way, $f_c$ is calculated as $1 - f_r$.

The correction factor $\gamma$ softens the effect of having a variable number of active communications in the system. A large number of them implies that we have to be stricter, whereas if there is a low number, we can allow higher overheads. In this way, we consider a regular distribution of communications depending on $t_r$. In other words, if the constraint is very hard or very soft, most of the nodes will be in hardware of software, decreasing the communication between both partitions. But if the constraint is medium, both partitions will be balanced, with a higher overhead, that we supposed to be the average number of existing communications. The factor $\gamma$ can be seen in Figure 2.

From our experimental work, and taking the previous considerations into account, we define the following relation, that measure the impact of a communication $c$ on the design:

$$\{ o(c) \cdot \gamma \cdot [n \cdot \pi(c)] \} > t_r$$

Notice that since $\pi$ is a factor relative to a node, and as we are comparing values similar in magnitude to $t_r$, it is necessary to weight this expression with $n = \text{Card}(N)$, the total number of nodes in the system.

Now, to find $\tau$ in an automatic way, the graph is traversed starting from the lowest communication, until we find one of them that meets the previous relation. This one is considered to be the first high communication worth to be grouped, and it is taken as the criterion $\tau$. So, all the higher communication are collapsed into groups, forcing them to remain inactive during the partitioning process.
7.- Experimental results.

The explained partitioning technique has been applied to the Codesign data-dependency graph that can be seen in Figure 3. Starting from an all-software solution, the partitioning process has been repeated 20 times, imposing a whole range of time constraints $t_r$, between the maximum and the minimum system times.

The obtained costs, that are a measure of the design quality, have been compared with those related to the classic FM approach. It can be seen in Figure 4 that the grouping technique clearly overcomes the standard one, as the results are always equal or better.

In Figure 5, the times are compared, depicting the grouping criterion chosen for every case. The time constraint is always met, and therefore, all the obtained results are valid. There are three remarks to make:

- The grouping technique offers better results when $t_r$ is high. That is because in this case, the number of nodes that are moved to hardware is low, as the constraint is easy to meet. Therefore, the algorithm performs few steps in which the taken decisions are critical. On the other hand, when moving a lot of nodes to hardware, the algorithm has more time to react, solving previous wrong decisions.
- The grouping criterion $\tau$ becomes more strict when the time constraint is harder. For intermediate values of $t_r$, $\tau$ remains stable. That is because there is a wide gap in the communications’ values, between those whose time overheads are 65 and 285.
- Although it is not shown, the time consumed by the algorithm to produce the results is much lower for the grouping technique than for the classic approach. That is because there are less groups in the former than in the latter.

8.- Conclusions and future work.

In this paper, a new partitioning method based on a grouping technique, has been presented. The most important advances are:

- Possibility of working with a complex estimation technique, allowing parallelism and hardware sharing.
- Extension of the definition of movement: There are now more possibilities to explore.
- Study of the grouping criterion selection, in order to avoid the FM algorithm’s weak points.

Talking about the future work, the main lines are:

- To study the repercussion on the presented technique when increasing the number of nodes in the system.
- To analyze more partitioning algorithms, and decide their suitability depending on the problem.

9.- References.