# Fast Design Space Exploration Method for Reconfigurable Architectures

Lilian Bossuet, Guy Gogniat, Jean-Luc Philippe
*LESTER Lab*
*University of South Brittany,*
*Lorient, France*
*{lilian.bossuet, guy.gogniat, jean-luc.philippe}@univ-ubs.fr*

## Abstract

*In this paper we propose an original and fast design space exploration method targeting reconfigurable architectures. This method takes place during the first steps of a design flow that works at the algorithmic level. It uses as input a high level specification of the application and is based on a functional model to describe the architectures to compare. This paper describes the projection step of the flow and presents an allocation heuristic that is based on communication costs reduction.*

## 1. Introduction

Actual systems on chip (SoC) need to improve in order to support new telecommunication and multimedia applications, particularly to reduce power consumption and to enhance flexibility. Integration and design evolutions are not sufficient to face these challenges. It is necessary to provide novel approaches that work at the system level to design more efficiently, and to target new technologies.

Reconfigurable architectures are becoming more attractive in terms of capacity, performances, low-power consumption and flexibility (through the possibilities of run-time reconfiguration and multi-granularity resources) [1]. They correspond to an efficient solution to the SoC challenge and will be unavoidable in a near future. But the design space of reconfigurable architectures is very large, because these architectures can be extremely heterogeneous in term of computing, memory and routing resources. Hence, it is very complex to find the best reconfigurable architecture for a panel of applications where each application can be dynamically configured on the architecture.

In order to help the designer it is necessary to develop tools that compare several architectures for different applications. We propose, in this paper, an original method of design space exploration for reconfigurable architectures that works at the algorithmic level.

The paper is organized as follows. Section 2 describes related work dealing with design space exploration methodologies. Section 3 presents our approach and section 4 details the projection step. Section 5 gives some results for different reconfigurable architectures. Finally, section 6 concludes the paper and exposes future direction.

## 2. Related work

Many research teams are focusing on reconfigurable architecture [1]. Some are working on design space exploration methodology in order to find the best architecture for a panel of applications.

Two ways are possible to explore reconfigurable architectures. The first one is to synthesize all the applications for the different target architectures, and to compare the overall performance results. In that case the results are very accurate, but it is necessary to have a specific synthesis tool for each architecture (which is not always available in the case of architecture exploration) or to use generic synthesis tools [2][3]. However, synthesis steps use very complex algorithms, which conducts to a limited and slow exploration. Furthermore it is necessary to have a very good knowledge of the target architectures when using generic synthesis tools since it is necessary to provide them a model of the target architectures. Hence, this method is not really adapted for a large and rapid architecture exploration and is more dedicated to do some architecture refinement steps.

The second way is to perform estimations. In that case it is necessary to consider a generic architecture model to describe the different architectures to target. The objective is to make relative performance estimations (speed, power consumption and area) in order to compare very quickly different architectures. Although the estimations do not give necessarily real and accurate performance results, it is enough to compare architectures since the important point in that case is that estimations are faithful and an absolute error is not the major concern.

Both exploration methods require having an architecture model. It is possible to consider a physical model. Then it is necessary to know precisely the physical parameters of the architecture (technology, routing type and size, routing switch resources, clusters size, etc). Versatile Place and Route (VPR) tool, developed at the Toronto University, is a very interesting approach that works on a physical model [2]. VPR is a synthesis tool that works at the logic level and is oriented for island style fine-grained architectures (as FPGA). It is not suitable for coarse-grained architectures. It is also possible to model architecture with a functional model. Each element of the architecture is described by the functions it can execute. The functional model enables to describe a large panel of architectures and the description are technological independent. This model is used in the generic place and route tool for fine-grained reconfigurable architectures called Madeo-Bet [3] that works at the logic level.

VPR and Madeot-Bet are not the only tools that use an architecture model, but there are very representative. Both are FPGAs oriented, but other approaches target reconfigurable architectures. In [4] the design space exploration flow targets mesh architecture called KressArray - a fast reconfigurable ALU. The exploration tool, Xplorer works at the algorithmic level and aims to assist the designer in finding a suitable architecture for a given set of applications. This tool is architecture-dependent, but the use of fuzzy logic to analyze the results of the exploration is a very attractive approach.

[5] presents the design space exploration for the Raw Microprocessor as an example of a tiled architecture. The Raw Microprocessor is reminiscent of coarse-grained FPGA and comprises a replicated set of tiles coupled together by a set of compiler orchestrated, pipelined, switches. Each tile contains a RISC-like processing core and SRAM memory for instructions and data.

## 3. Design space exploration flow

In this paper we proposed a design space exploration method targeting reconfigurable architectures. This method is based on an estimation approach and takes place at the very first steps of a design flow since it works at the algorithmic level. The objective of our work is to rapidly identify different architectures that present good performance and flexibility tradeoff. A functional model is used to describe targeted architectures. Our approach is flexible, in that it can be used to estimate performance for a wide variety of reconfigurable architectures. It is *fast*, in that estimates can be obtained without the time-consuming computation of programs such as Place & Route algorithms. Also, the model gives good *fidelity*; although there may be significant absolute errors in the performance estimation, the relative comparisons between two alternative architectures will be close to the relative errors that would be obtained after the synthesis steps.

### 3.1. Proposition of a Design Flow

The design exploration flow is depicted figure 1. The specification is provided in a high level language (C language) and is first translated into an intermediate representation - the HCDFG model. This model is a Hierarchical Control and Data Flow Graph allowing efficient algorithm characterization and exploration of complex applications including control flow and multi-dimensional data.

The first part of the flow (figure 1) is the **System Estimation** step [6][7], during which the application is characterized and scheduled. The results computed are defined as **Costs Profiles** i.e., scheduling for all the resources used by the application and for different time constraints. The available processing and memory resources to perform the scheduling are defined in a file called **User Abstract Rules**.

**Relative Estimation,** the second step of the flow (figure 1), aims to estimate the application performances on several reconfigurable architectures. Relative Estimation gives designer information to improve progressively the architecture definition with several runs in the design exploration flow.

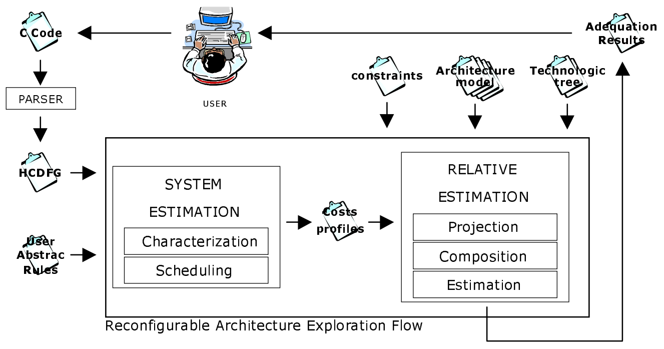

**Figure 1. Design Exploration Flow**

### 3.2 Specification

The application is specified with the C language. Once it has been functionally validated, it is translated into an intermediate model, which is a Hierarchical Control and Data Flow Graph (HCDFG) [8]. For sake of simplicity we can say that an application is modeled with several DFG connected through control structures, hierarchy and dependence relations. An important characteristic of the HCDFG model is that processing and data are explicitly represented with nodes in the graph. This decomposition enables to highlight the hierarchy and the potential parallelism of the application, which is an essential characteristic to perform an efficient algorithmic and architecture exploration.

### 3.3 System Estimation

System Estimation [6] consists in two steps; **Characterization** and **Scheduling**. During the Characterization step, the application orientation is analyzed through three axes - processing, control and memory. Specific metrics are used to find this orientation [7]. Once the application orientation has been exhibited the scheduling of the application is performed accordingly. For example, if the application is processing oriented, the processing resources are first scheduled and the memory resources are scheduled in a second step (the opposite if the application is memory oriented). The main objective of the System Estimation is to highlight the intrinsic processing and memory parallelisms of the application and to give some guidance on how to build the architecture (pipeline, parallelism, memory hierarchy etc.). Further details on the system estimation step are beyond the scope of this paper. Interested reader can refer to [6].

## 3.4 Relative Estimation

Relative Estimation is linked with the System Estimation through the cost profiles. The cost profiles describe the scheduling results (for all the computation and memory operations) for different time constraints. These information characterize the application to be implemented. In order to evaluate an application on different architectures, it is necessary at this level of the flow to specify the target reconfigurable architectures. For this, a functional model is used. Functional model is suitable for rapid relative comparison between architectures, with a high abstraction level.

With this model, called **HF model** [9], the architecture's elements are functionally described and the model allows representing different architectural styles and different architecture elements. However, the routing resources are not explicitly described, even if they are taken into account in the estimation tool. In fact, connection resources are taken into account by connection costs in the HF model. There are two types of elements in this model: the hierarchical elements and the functional elements. The hierarchical elements are used to describe the architecture's hierarchy. A hierarchical element can be composed of functional elements and other hierarchical elements. The functional elements are used to describe the architecture's resources. They can be logical, input/output, memory and computing resources. An extension of the HF model exists to model tile-based architectures, where the communications between the tiles must be explicitly modeled [10]. Figure 2 shows an example of architecture with several levels of granularity that can be modeled with the HF model.

Relative Estimation begins with the **Projection** step that makes the link between the necessary resources (processing and memory) of the application and the available resources of the architecture. Since, one of the

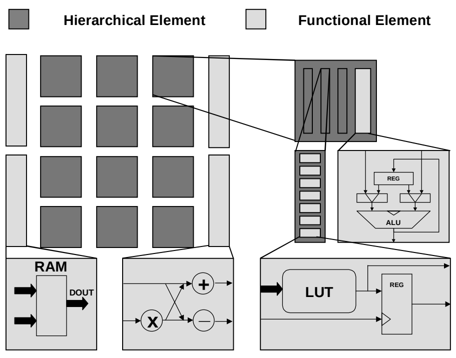

**Figure 2. Example of reconfigurable architecture could be specified with the HF**

major issues in reconfigurable architecture is the interconnection cost overhead we have defined an allocation heuristic based on communication costs reduction. The method for resources allocation is to consider the communications between all the processing and memory resources of the application, in order to assign in a same hierarchical level of the architecture, the resources that communicate the most.

If a difference of resource granularity exist between the necessary resources (from the application) and the available resources (in the architecture), the necessary resources can be decomposed as fine-grained resources using a library called **Technological Trees** (the same method is used with the PipeRench reconfigurable architecture [11]).

The **Composition** step takes into account the application's scheduling in order to refine previous estimations, since it is necessary to add resources dedicated to realize the scheduling like multiplexer, register or states machine. These additional resources are taken into account in the last step, the estimate process.

Finally, the **Estimation** step characterizes the application performances implemented into the modeled architecture. The estimations take into account the static costs of the model (interconnect costs between two hierarchical elements and the usage costs of the functional elements), and the dynamic costs of the application (critical path, operators communications, memories reads/writes). The results of this step are gathered into a file where the application is characterized for at least one modeled architecture. It is also possible to have several files for one application (and one architecture) in the case of architectural exploration, then one file is produced for each function's scheduling. The System Estimation step proposes several scheduling for a same application (so several parallelism levels), since for each potential time constraint a scheduling is defined. For example, if two time constraints are considered for an application then two result files are defined. This feature enables to explore the application's parallelism at a high level and to make a projection on a specific architecture.

## 4. Projection Algorithms

In this section we focus on the projection step. This step is particularly important in the flow since it has a strong impact on the final estimated performances. In the case of processing projection, the aim is to assign in a same hierarchical level of the architecture the resources that communicate the most. The hierarchical levels are described in the functional model of the architecture. In order to make this projection, the first step consists in the creation of a new graph with only treatments nodes and edges between nodes (the edges represent communications or data exchanges). This graph is reduced in order to take into account the scheduling (e.g. parallelism), the final graph is called Average Communications Graph (ACG).

## 4.1 Average Communications Graph (*ACG*)

The Average Communications Graph shows how each types of processing resources communicate with the other types of processing resources. This graph is used during the projection step to enhance the spatial locality of communicating resources. To build this graph it is necessary to have a processing graph of the application, i.e. a graph without memory node and control structures. In the ACG each edge represent the communications between two type of processing nodes.

Figure 3 shows on a simple example how to transform a processing graph in ACG. On the ACG, each node corresponds to a type of processing resources (adder, multiplier and subtracter).

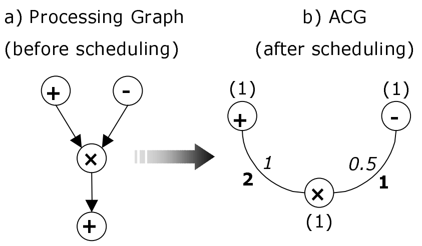

**Figure 3. Transformation of processing graph (a) in ACG (b)**

The number in brackets beside a node is the number of operators after the scheduling has be completed (see section 3.3). The boldface number beside an edge is the total number of communications between two processing types. The italic number beside an edge is the relative number of communications between two processing types. This value is obtained with the following expression;

$$RelativeComm_{Op1-Op2} = \frac{TotalComm_{Op1-Op2}}{NumberOp1 + NumberOp2}$$

Where *RelativeComm*$_{Op1\text{-}Op2}$ is the relative number of communications, *TotalComm*$_{Op1\text{-}Op2}$ is the total number of communications, *NumberOp1* and *NumberOp2* are the numbers of allocated operator of each type.

It is very fast to build the ACG from the HCDFG of the application. The ACG is the input point of the projection algorithms.

## 4.3 Generic projection algorithm

The projection step makes the link between the necessary (application) and the available (architecture) resources with the challenge that the most communicating resources must be assigned in a same hierarchical level of the architecture. Figure 4 shows the proposed algorithm. The algorithm begins with the ACG, and the first step searches in this graph the pair of nodes that exchange the most. In fact the step searches the edge with the highest relative value.

When a pair of nodes is determined it is necessary to know if the two types of processing nodes can be implemented by functional elements in a same hierarchical element (so it uses the architecture model). If the two nodes are compatible, they are assigned to the

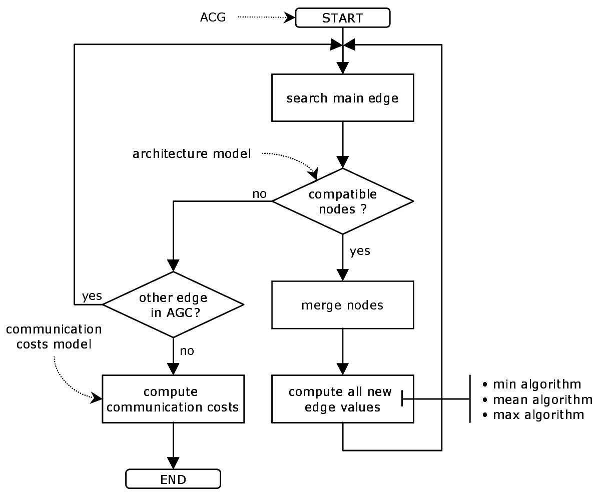

**Figure 4. Generic projection algorithm**

hierarchical element, and they form a new node, a composite node. This composite node has as parameter the processing types of the two previous nodes. The number of the two nodes is then decremented and all ACG edge values are re-computed.

To compute the new edge values, we use three algorithms that give respectively a lower and an upper bound and a mean value for the total communication costs. Since we work at the algorithmic level we do not target a specific synthesis tool. Hence it is important to give the designer some bounds that will give her (him) some guidance independently from any tool. Indeed, synthesis algorithms differ for two different tools (hence for two different architectures) so it is necessary to have an interval of result to guarantee the results' fidelity.

If, after the search of the main edge, the two nodes are not compatible, the search re-starts with other nodes of the ACG if they have still never been selected. If all the ACG edges have early been selected, and if any pair of nodes is compatible, then it is the end of the projection algorithm. Before, the communication costs are computed a communication costs model must be considered. The communication cost computation takes into account the resources hierarchical allocation, since a communication between two hierarchical elements does not represent the same cost than a communication in a single hierarchical element. The communication cost between two hierarchical elements and in a hierarchical element depends of the routing topology and the routing resources. Most details are given the section 4.7.

In the following sections 4.4, 4.5 and 4.6, we present the three allocation algorithms with a common single example. Figure 5 shows the transformation of a common ACG with the three algorithms. For each case, the considered main edge is between the multiplier node and the subtracter so in each case a composite node "multiplier – subtracter" is created (in the target architecture it is possible to assign in a same hierarchical element one operator multiplier and one operator subtracter). The number in the composite node represents the number of internal communications

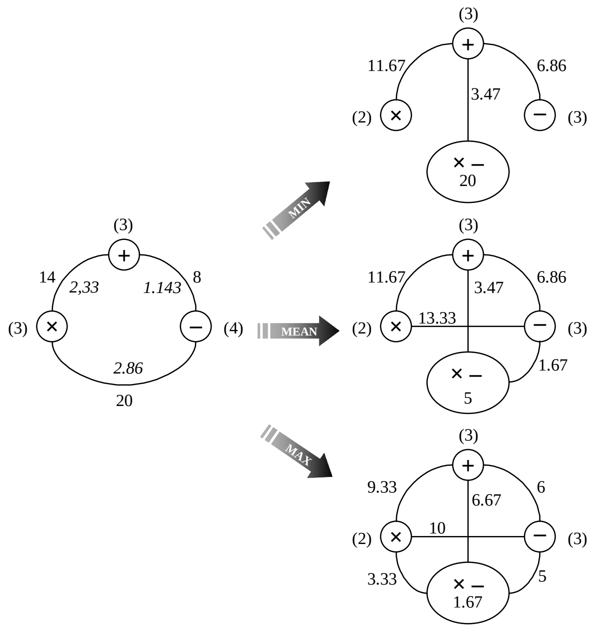

**Figure 5. ACG transformation for 3 cases, MIN, MEAN and MAX**

(communications in the hierarchical element between the two operators).

To understand the different algorithms, some notations are necessary:

- $N_\times$: number of multiplier in the node × of ACG
- $P_{\times,+}$: number of communication between node × and node + (e.g. edge value) of ACG
- $NewP_{\times,+}$: number of communication between node × and node + (e.g. edge value) of the new graph
- $Ci_{\times-}$: number of internal communication in the composite node ×-

## 4.4 Min algorithm (lower bound)

The strategy of the min algorithm is to consider that if two operators of distinct processing types can be assigned in a same hierarchical element, they will perform all the communications between all the nodes of the two processing types. So the number of internal communications of a new composite node (×-) is done by the following expression *(1)*. The edge between the two nodes (× and -) is deleted. If another node (+) has an edge with one of two nodes, or with the two nodes, then a new edge is created between this node (+) and the new composite node (×-) with the value done by the expression *(2)*.

$$Ci_{\times-} = P_{\times,-}$$

$$P_{+,\times-} = \frac{P_{+,\times}}{N_+ + N_\times} + \frac{P_{+,-}}{N_+ + N_-}$$

$$newP_{+,\times} = \frac{P_{+,\times} \times (N_+ + N_\times - 1)}{N_+ + N_\times}$$

Also, the edge values between the node (+) and the two others nodes (× and -) is done by the expression *(3)* for example for the node ×. With this algorithm most communications are executed in composite nodes. As the communication costs in composite nodes are low (because there is not communication across several hierarchical levels) the total cost of communication will represent a lower bound.

## 4.5 Max algorithm (upper bound)

The strategy of the max algorithm is to consider that all operators communicate uniformly. Hence, the number of internal communications of a new composite node (×-) is done by the following expression *(4)*:

$$Ci_{\times-} = \frac{P_{\times,-}}{N_\times \times N_-}$$

$$newP_{\times,-} = \frac{P_{\times,-} \times (N_\times - 1) \times (N_- - 1)}{N_\times \times N_-}$$

$$P_{\times,\times-} = \frac{P_{\times,-} \times (N_\times - 1)}{N_\times \times N_-}$$

The edge between the two nodes (× and -) is not deleted. But its value is changed as in the expression *(5)*. Each of the two nodes has a new edge with the new composite node, and the value is given by the expression *(6)* for the example for node ×.

If another node (+) has an edge with one of two nodes, or with the two nodes, then a new edge is created between this node (+) and the new composite node (×-) with the value given by the expression *(7)*. Also, the edge value between the node (+) and the two others nodes (× and -) is done by the expression *(8)* for example for node ×. With this algorithm the communications are uniformly executed between the different hierarchical elements, which corresponds to an upper bound. In that case the allocation algorithm do not take benefit from the architecture hierarchy, hence, the communication costs correspond to maximum costs.

$$P_{+,\times-} = \frac{P_{+,\times}}{N_\times}$$

$$newP_{+,\times} = \frac{P_{+,\times} \times (N_\times - 1)}{N_\times}$$

## 4.6 Mean algorithm (mean value)

The idea of the mean algorithm is to consider that two operators in a same hierarchical element must communicate more than two operators in two different hierarchical elements. The number of communication in a composite node, is the maximum communication number between two nodes. So the number of internal communications of a new composite node (×-) is done by the following expression *(9)*.

The new graph depends on the number of operator in the two considered nodes. For example if the number of operator × is bigger than the number of operator -, then a new edge is created between the node × and the composite node (×-), with as value the result of expression (10). If the two nodes have the same number of operators, any edge is created between the composite node and one of these two nodes.

$$Ci_{\times -} = MIN\left( \frac{P_{\times ,-}}{N_{\times}} ; \frac{P_{\times ,-}}{N_{-}} \right)$$

$$P_{\times ,\times -} = \frac{P_{\times ,-}}{N_{-}} - \frac{P_{\times ,-}}{N_{\times}}$$

$$newP_{\times ,-} = P_{\times ,-} - Ci_{\times} - P_{\times ,\times -}$$

The edge between the two nodes (× and -) is not deleted. But its value is changed by the expression (11). If another node (+) has an edge with one of the two nodes, or with the two nodes, then a new edge is created between this node (+) and the new composite node (×-) with the value given by the expression *(2)*. The edge value between the node (+) and the two others nodes (× and -) is done by the expression *(3)*.

### 4.7 Communication costs

On figure 4, we can see that the final step of the projection algorithm uses a communication costs model to compute the total communication cost. This model describes the costs to perform a data transfer in a hierarchical element and between several architecture hierarchical elements. These costs depend on the routing topologies (mesh, segmented base, hierarchical, etc) and on the routing resources (channel of wires, bus, crossbar, etc). A survey of interconnects architectures for reconfigurable architecture is done in [12]. In our approach these costs are relative since we want to compare two architecture styles instead of giving an absolute performance value. However, if the designer has a good knowledge of the target architecture, precisely costs can be considered.

Once the communication costs are determined, it is necessary to compute the number of communication between each hierarchical element in function of the graph result (at the end of one of the three previous algorithms). The total communication cost is given by the sum of the product of the communication number between two hierarchical elements and the cost for one of this communication.

## 5. Application and results

The one dimension Lee-DCT, a typical application of video compression [13], has been chosen to exhibit the exploration process and its ability to compare several architectures. Figure 6 presents the processing graph and the ACG for this application. The processing scheduling during the system estimation step has given four multipliers, four subtracters and three adders to perform for this application.

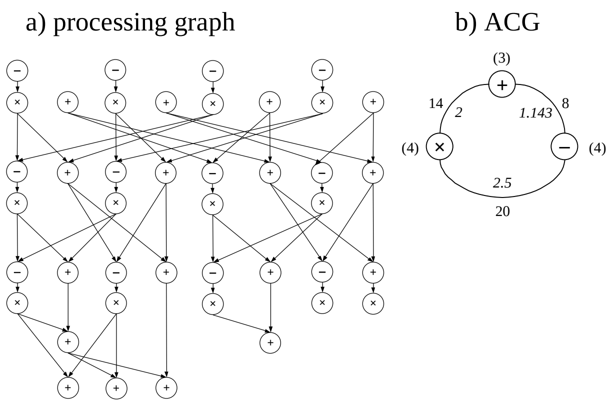

**Figure 6. Processing graph, and ACG for the 1-D Lee DCT**

Four architectures have been targeted during the projection step, with different hierarchical levels, and cluster sizes (number of functional elements by a hierarchical element). Figure 7 depicts it.

The results of the projection step is the total number of communications between resources weighted by the costs to communicate between hierarchical elements (communication costs model). For the considered

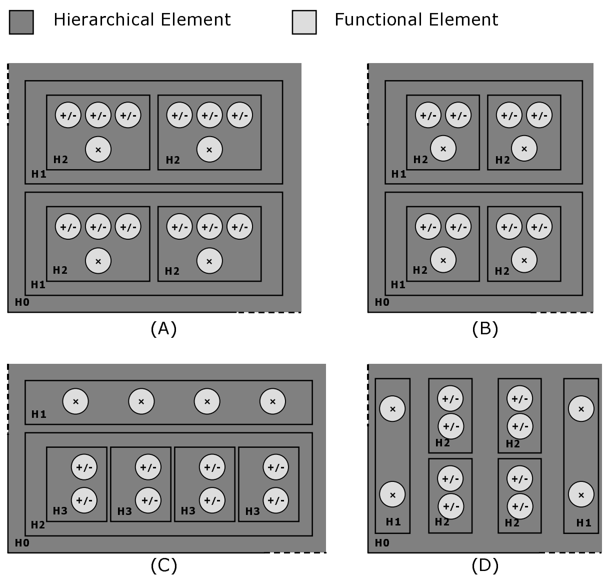

**Figure 7. Four examples of target architectures**

architectures the communication costs model is the following (for the example of architecture **A**):

- Cost of communication internal H2 = 0.1
- Cost of communication H2 - H2 = 0.2
- Cost of communication H2 - H1 = 0.3

Results are given in table 1. They highlight several interesting elements. Architectures **A** and **B** are better for this application than architectures **C** and **D**, since in the application the base structure is a MAC so it is necessary to have in a same hierarchical element the possibility to assign one multiplier, one adder and one subtracter.

We can notice that the results are close between the MIN and the MAX algorithm for the architecture **C** and **D**. Since these architectures have a limited exploration potential for this application, so the allocation will not have a strong impact on the final performances.

The last row of the table 1 gives the utilization rate of functional elements in the architecture. Although the **A** architecture is better to reduce communication costs, it has not a high functional element utilization rate. The **A** architecture is larger that the other architectures, so its static power consumption is higher than the other architectures. With the utilization rate information, we can see that **B** architecture is a good tradeoff between communication cost reduction vs. functional element utilization rate.

| architecture | a | b | c | d |
|---|---|---|---|---|
| MIN | 4.5 | 4.5 | 11 | 7.6 |
| MEAN | 6 | 6.5 | 11.3 | 7.9 |
| MAX | 9.2 | 9.45 | 11.6 | 8.2 |
| USE | 68.7% | 91.6% | 91.6% | 91.6% |

**Table 1. Results of projection algorithms**

In order to show the ability of our approach to make architecture exploration, we have computed the mean communication cost for architectures **A** and **B** with several numbers of hierarchical elements H2 in the hierarchical H1. We can see, on figure 8, that the larger is the cluster H1, the better is the communication cost. Indeed if H1 has many H2, local communications are promoted. This experiment show how is easy to explore some architecture characteristics.

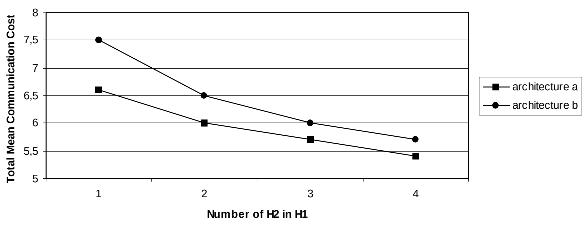

**Figure 8. Communication cost versus H1 size**

## 6. Conclusions and future work

In this paper we have presented a rapid allocation algorithm that takes into account the communications between application's operations. This algorithm uses the hierarchical view of the target architecture that is modeled with a functional model. Communication costs are taken into account with a communication cost model that depends on the routing structure in the target architecture. This algorithm is part of a design exploration flow that works at the algorithmic level and enables to compare quickly several reconfigurable architectures.

These approach which has been integrated in the codesign environment *Design Trotter* [6], enables to explore a large design space at an early stage of the design cycle and to characterize each solution in terms of area versus delay.

Future work will complete the design exploration flow development and test it on a tile-based architecture [10].

## 7. References

[1] R. Hartenstein. A Decade of Reconfigurable Computing : a Visionary Retrospective. *DATE'01, Munich, Germany, 13-16 March, 2001.*

[2] V. Betz, J. Rose, A. Marquart. *Architecture and CAD for Deep Submicron FPGAs.* Kluwer Academic Publishers, 1999.

[3] L. Lagadec, D. Lavenier, E. Fabiani, B. Pottier. Placing, Routing and Editing Virtual FPGAs. *FPL'01, August 2001.*

[4] U. Nageldinger. Coarse-Grained Reconfigurable Architecture Design Space exploration. *Ph.D. Thesis, University of Kaiserlautern, Germany, June 2001.*

[5] C. A. Moritz, D. Yeung, A. Agarwal. Exploring Optimal Cost-Performance designs for Raw Microprocessors. *FCCM'98, Napa, CA, USA, April 1998.*

[6] Y. Le Moullec, J.P. Diguet, J.L. Philippe. Design Trotter: a Multimedia Embedded Systems Design Space Exploration Tool. *In IEEE MMSP'02, Virgin Island, USA, 9-11 December, 2002.*

[7] Y. Le Moullec, N.Ben Amor, J.P. Diguet, M. Abid, J.L. Philippe. Multi-Granularity Metrics for the Era of Strongly Personalized SOCs. *DATE 03, Munich, Germany, 3-7 March, 2003.*

[8] J. P. Diguet, G. Gogniat, P. Danielo , M. Auguin, J.L Philippe. The SPF model. *FDL'00, Tübingen, Germany, September, 2000.*

[9] L. Bossuet, G. Gogniat, J.P. Diguet, J.L. Philippe. A Modeling Method for Reconfigurable Architecture. *IWSOC'02, Banff, Canada, July, 2002.*

[10] L. Bossuet, W. Burleson, G. Gogniat, V. Anand, A. Laffely, J.L. Philippe. Targeting Tiled Architectures in Design Exploration. *RAW'03, Nice, France, April, 2003.*

[11] M. Budiu, S. C. Goldstein. Fast Compilation for PipeRench Reconfigurable fabrics. *FPGA'99, Monterey, CA, USA, February, 1999.*

[12] H. Zhang, M. Wan, V. George, J. Rabaey. Interconnect Architecture Exploration for Low-Energy Reconfigurable Single-Chip DSPs. *IEEE Computer Society Workshop on VLSI, April 1999.*

[13] V. Bhaskaran, K. Konstantinides. *Image and Video Compression Standards : Algorithms and Architectures.* Kluwer Academic Publishers, 1995.