Architectural Synthesis with Interconnection Cost Control

JEGO Christophe, CASSEAU Emmanuel & MARTIN Eric

LESTER Laboratory, UBS University, France
Tel : (+33) 2.97.87.45.65 Fax : (+33) 2.97.87.45.00
E-mail: {First-name.Surname}@univ-ubs.fr http://lester.univ-ubs.fr:8080/

Abstract:

Architectural synthesis tools map algorithms to architectures under various constraints and quickly provide estimations of area and performance. However, these tools do not take the interconnection cost into account whereas it becomes predominant with the technology decrease and the application complexity increase. A way to control costly interconnections during the architectural process is presented in this paper.

Keywords:

Architectural synthesis, digital ASIC design, sub-micron technologies, interconnection cost

INTRODUCTION

Recent advances in VLSI technology lead to new design methodologies like architectural synthesis, so called behavioral synthesis. Architectural synthesis enables a significant productivity increase by raising the abstraction level of digital designs. This process, which explores the space of possible designs, reaches the "best" architectural solution satisfying a set of constraints such as propagation time, area or power dissipation. However,

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35498-9_57 both technology evolution and application complexity require models to be modified and algorithms that are used during the architectural synthesis process to be adapted. Actually, from the one hand, cost estimation models do not take into account the interconnections that are typically numerous with complex applications. On the other hand, interconnections have become cost effective and critical with deep sub-micron designs. In order to insure performance and reliability of the provided designs, the architectural synthesis flow has to be adapted.

This paper is structured in the following way: section 1 briefly presents the architectural synthesis flow and the GAUT behavioral synthesis tool. Section 2 introduces the interconnection cost problem with VLSI designs and high-level synthesis processes. Previous works about the interconnection cost problem in architectural synthesis tools are presented in section 3.1 and a way to reduce this cost is finally proposed in section 3.2.

1. ARCHITECTURAL SYNTHESIS OVERVIEW

Owing to recent advances in semiconductor technology, application complexity increase and time-to-market constraint, new design methodologies have to be developed. Architectural synthesis promises a significant productivity increase by raising the abstraction level of digital design. Basically, this process maps a behavioral description of an application into a register transfer (RT) level implementation. Since this process quickly provides area and performance estimations, it enables a more efficient exploration of the design space to be done.

1.1 Behavioral synthesis flow

From a behavioral description, an architectural synthesis tool generates an architecture of RT components such as arithmetic operators, registers, interconnection operators (multiplexors, demultiplexors and tristates) and memories, based on a target structural model (register, multiplexor or bus based architecture) [1]. The characteristics of the components (area, propagation time, power dissipation...) are initially given in a library.

Five steps/algorithms are involved during the architectural synthesis process: compilation, allocation, scheduling, binding, and resources optimisation [1,2]. Initially, a compilation step transforms the behavioral description into a control and/or a data flow graph representation. The allocation task determines the arithmetic operators to be used and their number, whereas the scheduling process assigns the flow graph operations to time intervals under a real time constraint. These two tasks are closely linked. If scheduling is performed before allocation, it imposes additional constraints on the operations with respect to allocation. Similarly, if allocation is performed before scheduling, it restricts the scheduling. That is the reason why it is difficult to characterise the quality of a given scheduling without considering the allocation step. Finally, binding maps variables and operations of the scheduled flow graph into the selected components. After the binding of operations to arithmetic operators and variables to storage components, additional algorithms are used for register sharing.

1.2 A behavioral synthesis tool : GAUT

The behavioral synthesis tool we use for this work is called GAUT. This tool has been developed by two French university laboratories: Lester (University of South Brittany, France) and Lasti (University of Rennes, France). GAUT is a pipeline architectural synthesis tool, which is dedicated to signal and image processing applications under real time execution constraints. From one behavioral specification, one mapping technology and one real time constraint, an optimised architecture is synthesised [3,4].

The generic architecture model is composed of four functional units: the processing unit, the control unit, the memory unit and the communication unit. The specification is written in VHDL, at a behavioral level without any architectural directive. After an algorithm compilation, the tool synthesises a data flow graph according to its generic model of architecture (register/multiplexor based architecture) and according to a library that contains the characteristics of components that come from previous logic/physical syntheses. GAUT starts the process with the processing unit synthesis because this unit undergoes the most important constraints for a real time application. Then the memory unit and the communication unit are generated. The control unit is described in order to be synthesised by a finite state machine design tool.

2. INTERCONNECTION COST PROBLEM

With the advancement of the VLSI circuit technology, a rapid scaling of the feature size has been performed. The minimum dimension of a transistor decreased from 2 μm in 1985 to 0.25 μm in 1999. According to the National Technology Roadmap for Semiconducteurs (NTRS) [5], it will further decrease at the rate of 0.7x per generation (consistent with Moore's Law) to reach 0.07 μm by 2010. Table 1 shows the evolution of the design integration features in CMOS technology since 1995 and gives the previsions until 2010.

	1995	1998	2001	2004	2007	2010
technology (μm)	0.35	0.25	0.18	0.13	0.1	0.07
supply voltage (V)	3.3	2.5	1.8	1.5	1.2	1
transistors per chip (M)	10	20	50	110	260	620
metal layers	4-5	5	5-6	6	7	7-8
ASIC area (mm^2)	450	660	750	900	1100	1400
frequency (Mhz)	300	450	600	800	1000	1100

Table 1: Evolution of the design integration features in CMOS technology.

Such scaling implies that the circuit performance will be increasingly determined by the interconnection performance : the wiring delay percentage relative to the cycle time actually becomes more important than the operator propagation time percentage [6,7]. For instance, interconnection contributes 50 percent of total delay in 0.35 μm and is expected to contribute up to 70 percent in 0.25 μm. Whereas this interconnection cost was not of great importance with technologies above 0.7 μm, interconnection design will play the most critical role in achieving of chips with sub-micron technologies.

New applications like multimedia or advance mobile communication systems require complex real time algorithm implementation under constraints such as area and/or power dissipation. Since behavioral synthesis tools map algorithms to architectures and provide fast estimations of area and propagation time, many different architectures can be rapidly explored according to the specified constraints. One of the high-level synthesis characteristics is an "optimal" reusing (sharing) of the operators and the registers. This reusing is performed with interconnection operators (multiplexors, demultiplexors and tristates) and involves interconnection cost (path delay and wiring area). However, the different steps of the synthesis process do not take into account the wiring area and unfortunately the path delay which are difficult to predict. Moreover, these steps are realised on the whole architecture without placement information, which leads to tremendous different wiring lengths when the synthesis of complex applications is concerned.

For instance, the architectural synthesis of the Viterbi algorithm, which is a typically complex application unlike usual synthesis examples like FIR filters etc., has been performed and logic and physical syntheses have been realised afterwards [8]. This work highlighted the problem of interconnection cost (wiring area and path delays) : a great difference may occur between the estimated characteristics of the architectural synthesis and the placed and routed architecture if interconnection cost (wiring) is not efficiently taken into account. In fact, the more complex the architecture is the higher estimation difference.

3. INTERCONNECTION COST CONTROL APPROACH

Since decisions made at the behavioral level may have a pronounced impact on the final design, estimations play a central role in guiding the process to optimal or near-optimal solutions. Typically, there are three major estimations used during the high-level synthesis flow: area, propagation time and power dissipation. These measures can be used at different levels of the process (for instance, they are used to drive the selection of the target architecture, to choose the library type, to select a particular component ...). Since all synthesis decisions depend on these estimations, their accuracy is essential for the generation of high-quality architectures.

Area and timing interconnection costs are known to be difficult to be accurately estimated especially when architecture becomes complex. However, even more than for a logic synthesis, routing performance may be critical for an architectural synthesis. Previous works about the interconnection cost problem in architectural synthesis tools are presented in section 3.1 and a way to control this cost is detailed in section 3.2.

3.1 Previous works

These last years, the interconnection cost problem at the architectural level was the subject of many publications [9-12]. These works are characterised by the techniques used to estimate the interconnection cost (wiring area and/or delay), the methods used to take into account these estimations at a behavioral level and the structural model used.

In [9], an initial solution is generated by partitioning the operations in the design to reduce the interconnection cost. Then a performance driven floorplanning is realised to provide an estimation of the interconnection cost. This estimation is then used for the scheduling of the operations. Afterwards, the solution is optimised by an iterative process that uses design transformations.

Xu [10] detailed a high-level synthesis flow which estimates the layout features with a specific estimation tool before performing the scheduling-binding task. The layout information is then used to guide the scheduling-binding task. The final result is evaluated without actually going through the time consuming phase of placement and routing. When time constraints are met, a structural RTL netlist and it is corresponding physical characteristics are generated.

The estimation of the interconnection length at the architectural level is one of the critical points. Mecha [11] presents a method that takes into account the interconnection wires during the behavioral synthesis without requiring floorplanning to be performed. It is an empirical method which formulates the interconnection length from a study of the routing rules features of Cadence CAD tools. Then, this length enables the interconnection delay to be estimated. In [12], the algorithm that evaluates the interconnection length does not require a complete placement of the components, it uses the topology of the connected components and the interconnections to estimate the distance between each pair of connected components [13]. The interconnection wire delays, computed from the estimated length of the wires, are finally included in an iterative behavioral synthesis.

However, these methods have various disadvantages: either they overestimate the interconnection cost and when the application becomes complex, the architectural solution can not satisfy the constraints (the allocation step selects too many components because the estimations are very pessimistic: worst case approach) or they are based on an iterative process (an initial architectural solution is improved from the estimation of its interconnections) therefore costly in CPU time. Our objective is to quickly provide reliable estimations of the architecture to the designer. We thus propose a different approach that enables the interconnection cost to be controlled all along the synthesis process and that takes care of costly interconnections. The interconnection cost control is carried out by the characterisation of the data flow graph variables. The synthesis also integrates a clustering task in order to insure temporal dependencies.

3.2 A way to control and reduce the interconnection cost

Architectural synthesis tools are based on a generic architecture model. This model is typically composed of four functional units : the processing unit, the control unit, the memory unit and the communication unit. The structural model of the processing unit is based on virtual elementary cells including an arithmetic operator, its connected registers and interconnection operators (figure 1).

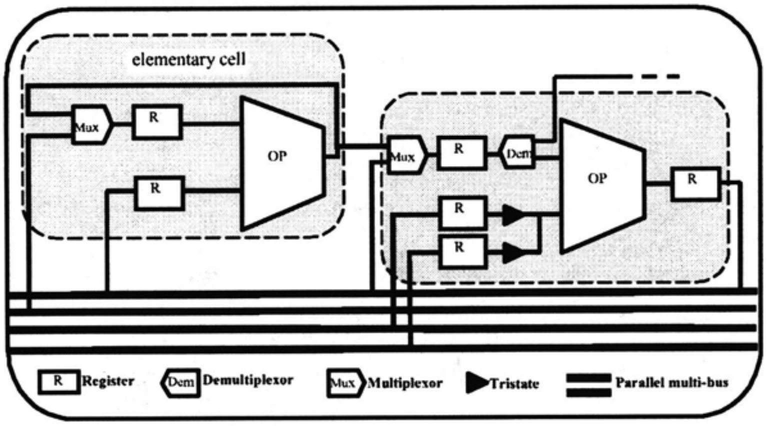

Figure 1: Structural model of the processing unit.

The arithmetic operators perform the data processing whereas the registers are used to temporarily memorise the variables and also to synchronise the data transfers between the processing unit and the memory or communication units. Interconnection operators are multiplexors, demultiplexors and tristates. The multiplexors and demultiplexors are necessary for the register reusing (optimisation step for register sharing) and the tristates make the control of arithmetic operator access possible. Finally, interconnection wires perform the data transfers into/between the elementary cells of the processing unit and a parallel multi-bus is used for the data communications between the four functional units of the architecture. Note that basically each variable of the algorithm is firstly recorded in a register. Different algorithms are next used for the decreasing of the register number during the optimisation step (optimisation by register sharing) : Left Edge [1], Branch&Bound [14], Branch&Bound with heuristics [15].

Consequently, a processing unit associated with this kind of typical structural model is composed of three different types of interconnection wires :

- local to an elementary cell : these interconnections perform the variable transfers into an elementary cell,
- local to the processing unit : these interconnections perform the variable transfers between elementary cells,
- global to the architecture : these interconnections perform the parallel multi-bus access and, by this way, the communication with the other functional units of the architecture.

In fact, the processing unit interconnection cost (wiring area and propagation delay) depends on the type of interconnection wires. On the one hand, this cost is low for interconnection wires that are local to an elementary cell or global to the architecture. On the other hand, for interconnection wires that are local to the processing unit, this cost depends on the complexity of the architecture (related to the algorithm complexity) and the place and route tool performance. Thus, in order to minimise the processing unit interconnection cost, the length and the number of interconnection wires that are local to the processing unit have to be minimised. Our objective is to firstly minimise their number all along the high-level synthesis process (selection-allocation, scheduling, binding and resource optimisation). Then their length will be controlled by a clustering step which has to be inserted between the binding and the register optimisation steps and which provides placement directives.

Since interconnections are associated with data transfers, the idea is to characterise the types of data (temporary processing data, constants or signals) and take advantage of their type. Three categories of data have thus been defined according to the interconnection features :

- category 1 : temporary processing data which are linked to a single arithmetic component,
- category 2 : temporary processing data which are linked to several arithmetic components,
- category 3 : temporary processing data and constants which are stored in the memory unit and input/output signals.

An initial characterisation of the variables is realised from the data flow graph associated with the behavioral description. In fact, in this step, the variables are associated with categories 2 or 3 because the set of arithmetic components is not yet being selected. This characterisation is presented figure 2a for a straightforward example.

The selection algorithm is a basic task of the architectural synthesis process that aims at optimising the cost of dedicated circuits. Its objective is to find the optimal set of components from a given library, for a behavioral description and a set of constraints. Different sets of components can be selected according to the library and the constraints. Then the allocation task determines the minimum number of every selected type of components. An initial automatic selection-allocation step that generates an "optimal" set of components in term of area under a time constraint is first performed. Consequently, the data characterisation may be refined : variables that are linked to one single arithmetic component are thus associated with category 1 whereas variables that are linked to more than one arithmetic component are still associated with category 2. By now, an optimisation phase is performed in order to minimise the number of variables associated with category 2 in favour of category 1 variables (related to component area and propagation time). The designer may take the opportunity of modifying the set of arithmetic components and/or their number to obtain a better set of components in term of data characterisation under a time constraint. Several data characterisations that correspond to different sets of selected components are carried out. The designer can thus choose one of the selection-allocation solutions according to the constraints (propagation time, data characterisation and components area). For example, three sets of components and their characterisation are presented in figure 2 b,c,d. They illustrate the evolution of the variable characterisation. In figure 2b, the architecture is costly in terms of interconnection wires because four variables are associated with category 2. The solutions that are proposed in figures 2c and 2d are more interesting from the interconnection cost point of view (only 2 variables are associated with category 2). However, solution d) is more restricted for temporal data dependencies. Furthermore, these data characterisations may change during the next steps of the synthesis process.

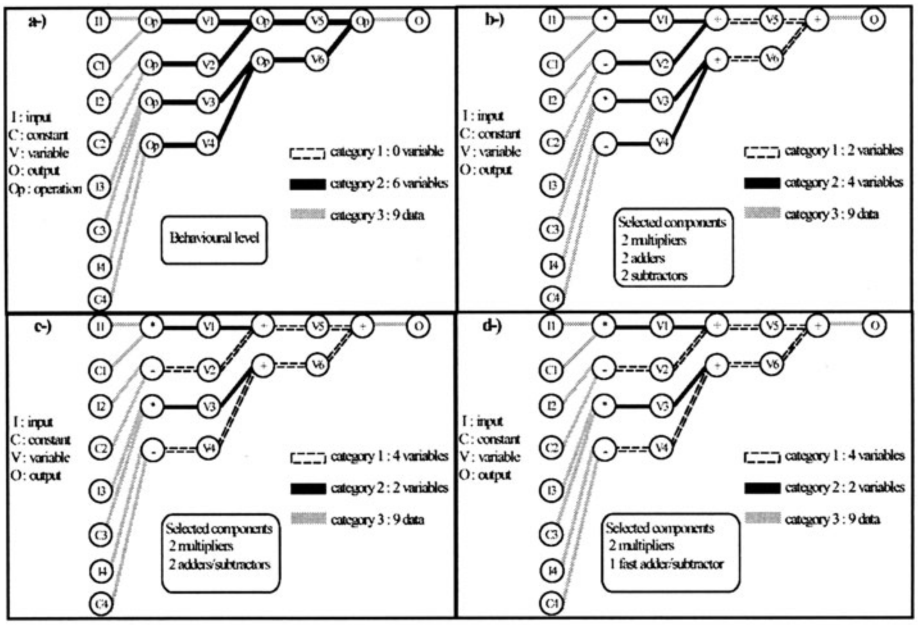

Figure 2: Characterisation of the data flow graph variables for the computation of $O = [(I_1*C_1)+(I_2-C_2)] + [(I_3*C_3)+(I_4-C_4)]$.

Many algorithms can perform the important task of scheduling. However, in order to take into account a given data characterisation, a resource constrained scheduling (List-Based Scheduling) [1] is used in this synthesis process. This algorithm is a generalisation of the ASAP algorithm with the inclusion of constraints. A scheduling priority list is provided according to a priority function. Naturally, the efficiency of this algorithm mainly depends on the priority function used. The priority function used in our approach depends on the mobility of the operations and the data characterisation constraints. For instance, the operations with a small mobility and associated with category 1 variables are scheduled in priority. By this way, the interconnection length is still shorten in the next step.

Finally, a binding algorithm is used to respectively assign the variables and operations of the data flow graph to registers and to the allocated components. Naturally, the variable characterisation is also taken into account in this step. For instance, during the previous steps, category 1 variables were variables linked to a single arithmetic component without regards to the number of this component (the operation performed by the component was only concerned). However, after the binding step, if a variable is linked to two mapped components (even if they carry out exactly the same operation), it becomes a category 2 variable, *i.e.* a potential costly interconnection. In this step, an operation is thus assigned to an available arithmetic component in such a way that the number of variables associated with category 2 is minimised in favour of category 1 variables.

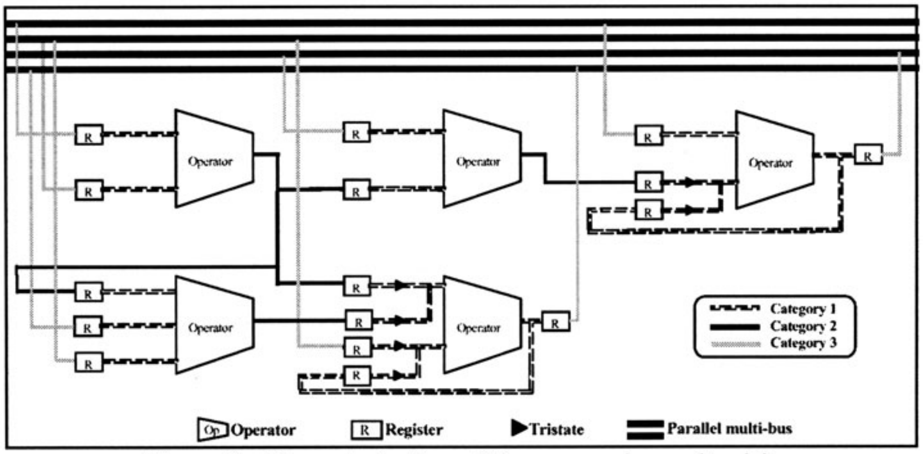

Figure 3: Characterisation of the processing unit wiring.

Obviously, any provided architectural solution will be composed of interconnection wires associated with category 2 variables, like the architecture presented in figure 3. For this reason, a clustering step is necessary to specifically control their cost. Since conventional place and route tools take hierarchical descriptions and placement directives into account, the generation of a hierarchical RTL description enables locally placed components, *i.e.* low cost interconnections. The clustering step of this synthesis flow thus consists in providing hierarchy in the RTL description. It starts with a non-partitioned set of components as provided by the binding process (for instance all the components of the processing unit of figure 3), and places them into clusters according to some component closeness measures. The processing unit is thus partitioned into elementary cells including an arithmetic component, its connected registers and interconnection operators (figure 1). The lengths of the wiring associated with category 1 variables can thus be minimised. Specific cluster placement directives are then provided in order to minimise the inter-cluster interconnection wiring, *i.e.* wiring associated with category 2 variables. Furthermore, when mobility of the operations is not critical, a register can be placed between two clusters to insure against wiring delay and to reduce the cluster placement constraints. Thus, one clock period is specifically dedicated to the data transfer between two clusters. This feature is actually carried out during the scheduling process.

With regards to the register sharing (optimisation step), the optimisation algorithm is applied to each distinct cluster neither on the whole processing unit. Figure 4 presents for instance the architecture obtained after the register optimisation step corresponding to figure 3. This method of register sharing is not as efficient as a register sharing applied to the whole processing unit from the number of registers point of view, however it insures local data transfers and, in this way, the time constraints to be observed.

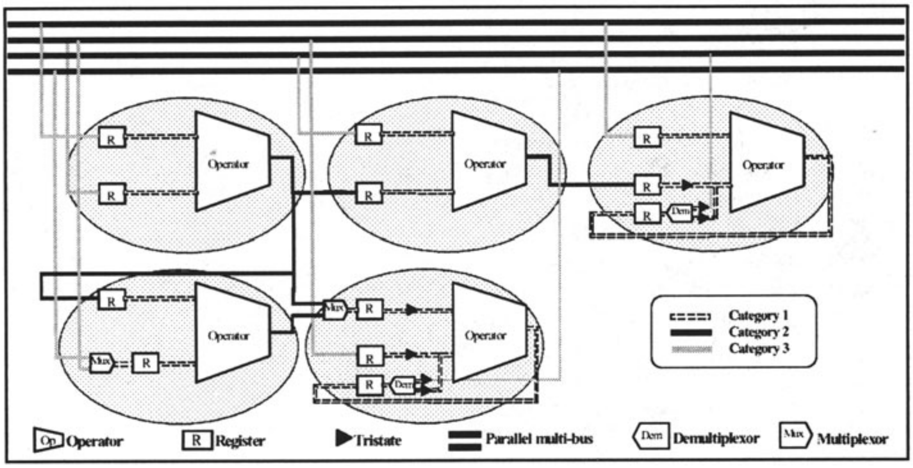

Figure 4: Processing unit obtained after clustering and register optimisation.

Naturally, this synthesis flow may involve a slight increase in the number of resources. However when complex applications are concerned and with sub-micron technologies, the additional resource area is often made up for the interconnection area decrease. Furthermore, the delay control, which is the most important point with VLSI design, can thus be significantly improved. This synthesis flow is currently being tested. The next step of our work is about the control unit, in particular the model of this unit. Actually, it seems that the critical path of the overall architecture is taken back from the processing unit to the control unit. A hierarchical finite state machine, included in the processing unit, may be a solution. This work will be reported later.

4. CONCLUSION

High-level synthesis is said to provide a significant productivity increase by raising the abstraction level of digital designs. Actually, HLS tools attempt to provide RT level design solutions with a quite good trade-off between cost and performance from a behavioral description. However, as interconnection cost becomes predominant, architectural synthesis tools have to take this additional cost into account. An approach that enables this cost to be controlled and that takes care of costly interconnections is proposed in this article.

5. REFERENCES

[1] D.D. Gajski, N. Dutt, A. Wu, S. Lin, "High-level Synthesis", Ed. Kluwer Academic Publisher 1992.

[2] M. C. FcFarland, A. C. Parker, R. Camposano, "The High-Level Synthesis of Digital Systems", In proceedings of IEEE, Vol.78, N°2, pp 301-318, 1990.

[3] E. Martin, O. Sentieys, H. Dubois, J.L. Philippe, "GAUT, an Architecture Synthesis Tool for Dedicated Signal Processors", In proceedings of EURO-DAC 93, pp. 14-19, 1993.

[4] J.L. Philippe, O. Sentieys, J.P. Diguet, E. Martin, " From digital signal processing specification to layout", In Logic and Architecture Synthesis : state-of-the-art and novel approaches, pp. 307-313, Chapman&Hall, 1995.

[5] Semiconductor Industry Association, "The National Technology Roadmap For Semiconductors", 1997.

[6] V. Moshnyaga, K. Tamaru, "Effect of Technology Scaling on Area-Delay Characteristics of RTL Designs: A Case Study", In proceedings of ED&TC'97, pp. 75-79, 1997.

[7] Wayne W.-M. Dai, "Chip Parasitic Extraction and Signal Integrity Verification", In proceedings of DAC 97, p 720-722, 1997.

[8] C. Jégo, E. Casseau, E. Martin, "Architectural Synthesis of a Complex Application : the Viterbi Algorithm", In User Forum proceedings of DATE 99, pp. 69-73, 1999.

[9] V. Moshnyaga, K. Tamaru, "A Floorplan Based Methodology for Data-Path Synthesis of Sub-Micron ASICs", IEICE Trans. on Information and Systems, pp. 1389-1395, Vol. E79-D, N° 10, 1996.

[10] M. Xu, F. J. Kurdahi, "Layout-Driven RTL Binding Techniques for High-Level Synthesis Using Accurate Estimators", ACM Trans. on Design Automation of Electronic Systems, pp. 312-343, Vol. 2, N° 4, 1997.

[11] H. Mecha, M. Fernandez, F. Tirado, J. Septién, D. Mozos, K. Olcoz, "A Method for Area Estimation of Data-Path in High-Level Synthesis", IEEE Trans. on CAD of Integrated Circuits and Systems, pp. 258-265, Vol 15, 1996.

[12] J. Hallberg, Z. Peng, "Estimation and Consideration of Interconnection Delays during High-Level Synthesis", In proceedings of Euromicro'98, pp. 349-356, 1998.

[13] A. Alvandpour and C Svensson, "A Wire Capacitance Estimation Technique for Power Consuming Interconnections at High Levels of Abstraction", In proceedings of Patmos'97, pp. 305-314, 1997.

[14] W. Grass, "A Branch-and-Bound Method for Optimal Transformation of Data Flow Graphs for Observing Hardware Constraints", In proceedings of EDAC 91, pp.73-77, 1991.

[15] J. Septién, D. Mozos, F. Tirado, R. Hermida, M. Fernández, "Heuristics for Branch-and-Bound Global Allocation", In proceedings of EURO-DAC 92, pp. 334-340, 1992.