A Horizontal Optimization Algorithm
for Data Path/Control Synthesis

by

Zebo Peng

This paper was published as a part of the Proceedings of the
1988 IEEE International Symposium on Circuits and Systems,
Helsinki University of Technology, Finland, June 7-9, 1988

RESEARCH REPORT
CADLAB, September 1988

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden
The Department of Computer and Information Science
Linköping University

PhD theses:

* Available at: University of Microfilms Intl., 300 N. Zeeb Road, Ann Arbor, MI 48106, USA.

( Linköping Studies in Science and Technology. Dissertations. )

No 97 Andrzej Lingas: Advances in Minimum Weight Triangulation, 1983.
*No 109 Peter Fritzson: Towards a Distributed Programming Environment based on Incremental Compilation, 1984.

Licentiate of engineering Theses:

( Linköping Studies in Science and Technology. Theses. )

No 73 Ola Strömforss: A Structure Editor for Documents and Programs. 1986.
No 113 Ralph Rönnquist: Network and Lattice Based Approaches to the Representation of Knowledge. 1987.
No 118 Mariam Kamkar, Nahid Shahmehri: Affect-Chaining in Program Flow Analysis Applied to Queries of Programs. 1987
No 126 Dan Strömberg: Transfer and Distribution of Application Programs. 1987
A Horizontal Optimization Algorithm for Data Path/Control Synthesis

by

Zebo Peng
E-mail: spe@ida.liu.se

Abstract: This paper describes a new approach to register level synthesis of data path/control from high level behavioral descriptions. This approach utilizes a horizontal optimization algorithm which makes design decisions concerning operation scheduling, data path allocation and control allocation at the same time. The proposed algorithm works iteratively on parts of the design space selected according to their relationship to the current performance critical path. Experimental results show that the horizontal approach produces better results in respect to performance/cost than a vertical one that separates operation scheduling, data path allocation and control allocation into three subprocesses.

This research has been supported by the Swedish National Board for Technical Development (STU).

This paper was published as a part of the Proceedings of the 1988 IEEE International Symposium on Circuits and Systems, Helsinki University of Technology, Finland, June 7-9, 1988

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden
A Horizontal Optimization Algorithm for Data Path/Control Synthesis

Zebo Peng

1. INTRODUCTION

Synthesis of VLSI systems from their high level behavioral descriptions into register transfer level structures consists of making design decisions concerning partitioning the systems into clusters of operations, scheduling these operations, allocating data path to perform the operations, generating their control circuits, etc, which are a set of highly interdependent problems. It is impossible to give a global optimal solution to all of these problems for a practical VLSI design. To address the complexity problem, the synthesis process is usually partitioned into a sequence of subprocesses, namely, global optimization, operation scheduling, data path allocation, control allocation, module binding, etc. These subprocesses are then performed one by one in some order with possible iterations [3], [8].

One of the problems with such a vertical approach is that optimization trade-offs between different subprocesses can not be made systematically. For example, if the data path allocation is done first and during the control allocation process it is found that some design constraints can not be satisfied unless the data path be reconfigured, either very expansive back-tracking facility (due to the complex design data structure) must be provided or the synthesis process has to be restarted from the beginning [4]. Most of the behavior entry synthesis systems which use a vertical approach have concentrated on the synthesis of general-purpose computer architectures [1], [3], [4], [8], where the data path allocation problem dominates the synthesis process: Consequently, the integration of the control allocation problem into the data path design phase is not critical.

We have attempted to address the design of systems for a wide class of applications from CPU based architectures to special hardwares where the control allocation is at least as important as the data path design. We propose a horizontal way to design such systems; instead of partitioning the synthesis process, we integrate them together. That is, the synthesis algorithm performs operation scheduling, data path allocation and control allocation at the same time. To reduce the complexity, the designed system is partitioned into a set of small parts. The synthesis algorithm works iteratively on one or several small parts of the designed system at a time. To guarantee global optimization, we need a systematic way to select parts to work on at each synthesis step. This is accomplished by a critical path analysis procedure. Experimental results show that the proposed horizontal
algorithm produces better results in respect to performance/cost than a vertical approach that separates operation scheduling, data path allocation and control allocation into three subprocesses.

In the next section, we will describe briefly the overall structure of a VLSI design automation environment where the horizontal optimization algorithm is used. Section 3 describes in detail the optimization algorithm and its implementation. Some design examples are then discussed in section 4 together with the experimental results and comparisons between the proposed horizontal approach and other approaches. Finally we summarize the discussions in section 5. This research project is financially supported by grants from the Swedish National Board for Technical Development (STU).

2. OVERVIEW OF THE SYSTEM

The proposed horizontal optimization algorithm has been designed and implemented in the CAMAD (Computer Aided Modelling, Analysis and Design of VLSI systems) system [5]. CAMAD is a register transfer level synthesis tool built around a unified design representation which consists of separate but related models of control and data path. The control structure is modelled as a timed Petri net ([7]) with restricted transition firing rules. The data path, on the other hand, is represented as a directed graph. A high level behavioral description can directly be translated into such an extended timed Petri net (ETPN) representation which becomes the preliminary structural description of the implementation. This preliminary implementation structure is usually a very expensive and complicated one. However, once this is done, the synthesis of register transfer design from a behavioral description has become a problem to transform a bad implementation into a good one, or an optimal one. This later process can be done automatically with a horizontal approach.

The overall structure of CAMAD is illustrated in Fig.1. The input to CAMAD is a PASCAL program together with a set of design constrains provided by the designers. The PASCAL program is used to represent the functional behaviors of the VLSI chip. This behavioral description is translated into the ETPN representation. The ETPN representation serves then as the intermediate representation of the design and will be manipulated by a set of basic synthesis transformation algorithms. These algorithms manipulate both the data path and the control structure to reflect design decisions made about operation scheduling and resources allocation (i.e., data path allocation and control allocation). The horizontal optimization algorithm decides the order of these basic synthesis transformations so as to move the preliminary design towards the optimal or near optimal one.

Besides the essential synthesis algorithms, CAMAD includes also other tools for system level design tasks as shown in Fig.1. For example, the module partitioning algorithm
takes an ETPN design representation and partitions it into a set of asynchronous modules which can operate at different rates of speeds [5]. The microprogram generation algorithm, on the other hand, produces automatically a microprogram and its execution mechanism from the control description in ETPN forms.

3. THE OPTIMIZATION ALGORITHM

The kernel of the synthesis process in the CAMAD system is to optimize the preliminary implementation structure which comes directly from the translation of the behavioral description. This synthesis process is carried out by a set of basic synthesis transformations [6]. A synthesis transformation usually deals with only a small set of minor parts of the design but it makes design decision concerning operation scheduling, data path allocation and control allocation at the same time. The basic trade-offs for the design decisions are between performance and cost. To reduce the cost, the design tries to share resources as much as possible, which is done by resource merger transformations.

For example, an Operator Merger folds two operator nodes into one, which represents the decision to share an operator by two operations. Such a merger usually results in the
reduction of a major component but addition of some minor ones (e.g., introduction of new multiplex or expansion of the existing multiplex as well as control circuits). This usually means that the total cost of the design decreases. However, the introduction or the expansion of the multiplex may increase the time required to finish some operations. Consequently, it could affect the performance of the system and result in the violation of certain design constraint. CAMAD provides algorithms to efficiently analyze this effect so as to make the decision whether the merger will be performed.

The global effect of such transformations, on the other hand, is much more difficult to predict. The choice of a sequence of basic transformation algorithms to apply to a ETPN representation affects the final result of the design; one transformation may, for example, prevent another one with more effective gains. A systematic way to reach the optimal solution is to search for all possible sequences and compare their final results; the best will then be selected, which is a NP complete problem. A heuristic design space search strategy is needed to achieve an optimal or near optimal solution for practical problems.

The general strategy we use is an iterative improvement strategy. The criteria used for the choice of subparts to be improved are the critical path for performance. A critical path is the path in the control Petri net which affects the overall performance the most. The performance criteria are usually specified in respect to the time required for a token to come from one Petri net place to another (from the initial place to the final place if the algorithm is defined as a big loop). Under this definition, what we measure is the dynamic behaviors of the system, not just the static structure of the control Petri net.

The horizontal optimization algorithm is as follows:

1) Calculate required time $k_i$ for each place $P_i$ in the control Petri net;

2) Calculate the minimal clock cycle time $C_i$;
   IF NOT ($C_t$ satisfies clock requirement) THEN
   BEGIN
     Find $P_i$ where $k_i = C_t$;
     Split $P_i$;
     Repeat Step 2;
   END;

3) UNTIL Satisfied DO
   BEGIN
     Find critical path $C_P$ in the control Petri net;
     IF $C_P$ is the same as the old critical path THEN exit;
     IF $C_P$ satisfies performance requirement THEN
     Apply basic transformations in respect to the data path units associated with $C_P$,
     taking performance as first priority;
     ELSE
     IF $C_P$ has places which can be parallelized THEN
BEGIN
  Parallelize the places;
  Mark the places as unstretchable;
  Repeat Step 3;
END
ELSE
BEGIN
  Find $P_i$ where $i = C_t$;
  IF $P_i$ cannot be split THEN exit;
  Split $P_i$;
  Goto Step 2;
END;
Apply basic transformations in respect to data path units in the order of similarity to the data path units in the critical path, taking cost as first priority;
Apply basic transformations in respect to control places, taking cost as first priority;
Design a preliminary floorplan;
Calculate new Cost and Performance;
Satisfied := (Cost satisfies Cost-Constraint) AND (Performance satisfies Performance-Constraint)
END;

The first step of the horizontal optimization algorithm is to calculate the time required for the set of operations associated with each Petri net place. This information is then used to determine the minimal clock cycle time $C_t$. $C_t$ equals the time required for the most time-consuming operation set. If this minimal clock cycle time is longer than the maximal allowed clock cycle time (if given), the most time-consuming operation set will be split up into a sequence of more primitive ones each of which takes approximately the same time as the given clock cycle. Both step one and two take linear time with respect to the number of the places in the Petri net and the number of data path units in the data path.

The major optimization is performed in the third step of the algorithm which repeats the following tasks: first find the critical path in the control Petri net. The part of design dealing with data path units associated with the critical path will then be done. Basic transformations of the design representation are carried out with performance taken as first priority. In this way, the performance of the design will not be affected by, for example, sharing of resources.

If the critical path does not satisfy the performance requirement given by the designers, the set of the places in the critical path will be checked to see whether two of them can be parallelized, i.e., their associated operations will be done in parallel instead of in sequence. If this is the case, the parallelization will be carried out, and the algorithm
returns to the beginning of step 3. A new critical path will be found and the design process continues. If no places can be put into parallel form, the only way to satisfy the performance requirement is to increase the clock frequency. For this purpose, the set of operations associated with the most time-consuming place will be split up into two. The algorithm then goes to step two to calculate the new clock cycle time.

After the design concerning the current critical path is done, the rest of the design will be performed in the order of the similarity to the data path units in the critical path. This time the cost will be taken as first priority when making cost/performance trade-offs. That is, the operations associated with the off-critical path will try to share resources as much as possible. Design decisions about off-critical path places in the control Petri net will also be made to reduce the complexity of the control function. These transformations of the design may, of course, result in new critical paths; therefore the above design process will repeat until a satisfactory design has been reached or the new critical path equals the old one, which means that no more optimization can be done.

To ensure that the optimization algorithm will not alternate between two critical paths and never arrive at a solution, we have used the following two methods. First, every time two places are parallelized, they are marked as unstretchable. Thus, the situation that two off-critical-path places are stretched to create a new critical path and then parallelized so that the original critical path becomes again the critical path will not occur.

The second situation which may result in an endless loop is that after a place with maximal time is split into two sequential ones, they may later be folded into the original place. This situation is excluded by carrying out folding transformations only if they will not result in a place whose required time is longer than the current clock cycle time (i.e., the time required for the most time-consuming operation set). Consequently, the clock cycle time either remains the same or decreases from one iteration to another during the process of optimization; therefore, the above dead loop situation cannot occur. Since there are not any other situations which may result in dead loop, the optimization algorithm always terminate.

In general, as the horizontal optimization algorithm depends upon a sequence of local transformations, it is not guaranteed that it will generate a globally optimal solution. Nevertheless, the critical path analysis procedure takes into account the global aspect of the design. Therefore, the critical-path-first method we used usually generate quite good solutions, especially in the case where the performance is dominated by a single critical path. In the situation where several paths affect the performance evenly, the following two techniques are used to increase the chance of reaching an optimal solution.

The first technique used is to guide the off-critical path transformations by analyzing the secondly-critical path, then the thirdly-critical path, and so on [6]. The second technique
is to introduce some randomness into the algorithm. When several transformations are simultaneously applicable and no good rule can be used to choose one, a random choice is made. In this way, the algorithm can be used to generate several solutions and select the best among them.

4. EXPERIMENTAL RESULTS AND ANALYSIS

We have run several designs on CAMAD to test the performance of the horizontal optimization algorithms. Three of the examples are discussed here. The first example is the design of a small microprocessor with a simple instruction set. The second example is a special interface controller which implement an asynchronous/synchronous conversion function. The third example is to implement a square root algorithm directly on silicon which takes as input an integer value and produces as output the square root of the input integer.

The design examples we have used here show that CAMAD can handle quite a wide spectrum of designs. This is very different from most of the previous work done on the problem of synthesizing behavioral description into register level structures which concentrated on a small set of applications dominated by either data path consideration or control structure implementation [1], [3], [8]. Consequently, it is very difficult to compare the results produced by CAMAD with other design systems which focus on a much smaller set of problems. Further, different approaches use quite different assumptions about the underlying model of hardware implementation for the back end synthesis, for example, the module libraries of different systems may have quite different sets of primitive components with different parameters. Consequently, a direct comparison of results produced by different design synthesis systems could be misleading.

We have, therefore, tried to make a more rigid comparison. We compare the results of the proposed horizontal approach to the results of using a vertical approach under the same environment with the same examples. As the ETPN is used as the unified design representation, the implementation of a vertical approach in CAMAD is quite straightforward (which also demonstrates one of the features of CAMAD, namely, it provides a framework for the development of a set of different design algorithms). We have implemented a vertical approach which makes design decisions first about operation scheduling and then about data path allocation and finally about control allocation. During each subprocess optimization is performed on the basis of both cost and performance criteria as it is done in the horizontal algorithm.

Comparisons of the results produced by the two approaches are summarized in Fig.2, where the cost is measured in terms of the estimated area of the data path layouts without the pads (based on the feature size). Performance, on the other hand, is measured by the average time (in microseconds) required to finish a typical set of
Solid dots produced by the horizontal algorithm; Hollow dots produced by the corresponding vertical one.

Fig. 2 Horizontal approach vs. vertical approach

operations. In the microprocessor example it measures the average time to execute an instruction; in the interface controller example the average time to complete a data conversion operation; and in the last example the average time to compute a square root.

The present approach differs mostly from the other in that it integrates the operation scheduling, control allocation and data path allocation together in the synthesis process. This makes it possible to make systematic design trade-offs between control structure and data path. As such, in the synthesis of the interface controller where the control structure is tightly connected to the data path organization, the vertical approach shows a definite advantage over the vertical one. For the design of the conventional microprocessor architecture, it does not make too much difference because the data path allocation almost dominates the design.

The second important properties of the present approach is that back-tracking is not necessary. In practical design cases, the design data structure like ETPN is usually very complicated, which means that if you want to implement a back-tracking mechanism, it will be very expansive. In CAMAD the ETPN works as a central design data base which needs to maintain only one copy of a design at a time. This design representation is
constantly updated by a sequence of synthesis transformations. In this manner, the effects of synthesis are immediately visible to the other design algorithms. In this sense, this approach is similar to the expert system approach [2]. In the approach proposed in [2] the design data structure is analyzed in order to find a transformation sequence (a set of expert design rules) with some depth before the transformation sequence is executed. In our approach, however, we execute each transformation (a design rule) immediately once it is found. This is possible in CAMAD because the effect of a transformation may be later overridden by some other transformations, if it does not lead to a promising result.

In our approach, most of the basic synthesis transformations of the ETPN design representation deal with only a small set of minor subparts of a design at a time. The accumulative effects of a sequence of such small changes may result in extensive reorganization of the design. For example, a bus may be generated by slowly grouping a set of arcs together. Nevertheless, more radical design reconfigurations that require overall reorganization of the design structure cannot be achieved this way. For example, CAMAD will not be able to automatically make a decision to introduce a pipeline implementation structure into the design.

5. CONCLUSIONS

We have described a new approach to the synthesis of VLSI systems from their high level behavioral descriptions into structural implementations. This approach integrates operation scheduling, data path allocation and control allocation of a traditional synthesis process. Therefore, the advantages from compacting the data path, for example, can be compared immediately with the possible expansion of the control logic to justify the compaction.

The horizontal algorithm is designed to make decisions on what should be done in each synthesis step. When making such design decisions, it makes trade-offs between data path and control, resource sharing and operation scheduling, as well as cost and performance. The experimental results show that under the same conditions, a horizontal approach performs better than a vertical approach in respect to overall system performance and estimated layout area.

6. REFERENCES


A Horizontal Optimization Algorithm for Data Path/Control Synthesis

Zebo Peng

Abstract: This paper describes a new approach to register level synthesis of data path/control from high level behavioral descriptions. This approach utilizes a horizontal optimization algorithm which makes design decisions concerning operation scheduling, data path allocation and control allocation at the same time. The proposed algorithm works iteratively on parts of the design space selected according to their relationship to the current performance critical path. Experimental results show that the horizontal approach produces better results in respect to performance/cost than a vertical one that separates operation scheduling, data path allocation and control allocation into three subprocesses.
A Selection of Previous Research Reports.


LiTH-IDA-R-88-18 Nils Dahlbäck: Mental Models and Text Understanding - a Commented Review.


LiTH-IDA-R-88-16 Lin Padgham: A Model and Representation for Type Information and Its Use in Reasoning with Defaults. Also in Proc. of AAAI’83, American Association for Artificial Intelligence, 1988.


LiTH-IDA-R-88-13 Erik Tengvall: Ett kartorienterat planeringsystem för autonom farkoster, en design diskussion.


LiTH-IDA-R-88-11 Mats Wirén: An Incremental Chart Parser for PATR.


LiTH-IDA-R-88-09 Peter Fritzson: Incremental Symbol Processing.


LiTH-IDA-R-88-06 Christer Bäckström: A Representation of Coordinated Actions Characterized by Interval Valuated Conditions.

LiTH-IDA-R-88-05 Christer Bäckström: Keeping and Forcing: How to Represent Cooperating Actions.


LiTH-IDA-R-88-02 Ulf Nilsson: Inferring Restricted AND-Parallelism in Logic Programs using Abstract Interpretation.
The Department of Computer and Information Science
Linköping University

organizes undergraduate and graduate studies in Computer Science, Telecommunication and Computer Systems, and Administrative Data Processing. Research activities have an emphasis on advanced software technology and computer systems design and are organized in a number of research laboratories:

- **ACTLAB - Laboratory for Complexity of Algorithms**, which is concerned with the design and analysis of efficient sequential and parallel algorithms, and complexity theory, especially in the areas of computational geometry, data structures on bounded domains and graph algorithms.

- **ASLAB - Application Systems Laboratory**, which studies design of advanced support systems for interactive use of computers, including tools for automated construction of applications software.

- **CADLAB - Laboratory for Computer-Aided Design of Electronics**, which concentrates its research activities around tools for integrated development of hardware and software, graphics-based modelling and simulation techniques.

- **LIBLAB - Laboratory for Library and Information Science**, which studies methods for access to documents and the information contained in the documents, concentrating on catalogs and bibliographic representations, and on the human factors of library use.

- **LOGPRO - Laboratory for Logic Programming**, which concentrates its research activities around foundations of logic programming, relations to other programming paradigms and methodology.

- **NLPLAB - Natural Language Processing Laboratory**, which conducts research related to the development and use of natural language interfaces to computer software.

- **PELAB - Programming Environments Laboratory**, which works with design of tools for software development, specific functions in such tools and theoretical aspects of programs under construction.

- **RKLLAB - Laboratory for Representation of Knowledge in Logic**, which covers issues and techniques such as non-monotonic logic, probabilistic logic, modal logic and truth maintenance algorithms and systems.

**Research Reports 1988**


LiTH-IDA-R-88-34  Arne Jönsson, Nils Dahlbäck: Talking to a computer is not like talking to your best friend. Also in *Proc. of the First Scandinavian Conference on Artificial Intelligence*, March 9-10, 1988, Tromsö, Norway.


