31

Synthesis : From Digital Signal Processing Specifications to Layout

PHILIPPE J.- L.[1], SENTIEYS O.[1], DIGUET J.-P.[1], MARTIN E.[2]
[1]Laboratoire d'Analyse des Systèmes de Traitement de l'Information, ENSSAT 6 rue de Kerampont 22300 LANNION FRANCE
Tel: (33)/96/46/50/30; FAX: (33)/96/37/01/99;
Email Name@merlin.enssat.fr.
[2]IUP LORIENT 10 rue Jean ZAY 56100 LORIENT FRANCE
Tel: (33)/97/87/28/61; FAX: (33)/97/87/28/15;

Abstract

To be efficient, a design methodology has to take into account the following user requirements: fast prototyping in order to satisfy the time to market constraint, the production of efficient architecture, the comparison of the complexity of different DSP algorithmic solutions. These requirements may be associated to user constraints such as real time processing, cost or power consumption, the re-use of previous developments.
Such a design methodology has been developed and integrated into the tool GAUT. From the VHDL behavioural specification of a DSP algorithm, a time constraints and a technologic library (FPGA, standard cells), the dedicated pipeline architecture is synthesized. This VHDL description is then computed by a logic synthesis tool. This design flow, which produce quickly efficient dedicated architectures, is validated on some digital signal processing applications like acoustic echo cancellation and ADPCM.

Keywords

High-Level Synthesis

1 INTRODUCTION

The implementation cycle of an application has to consider three main points: the architecture, the algorithm, and the characteristics of the application itselve.

For one algorithm, you have to select which architecture you need. If the architecture already exists, you have to decide which part of the algorithm you have to implement on which part of the architecture. If the architecture doesn't exist, you have to design it in respect with the algorithm you have selected for the application. This one is very often: time, consumption, or cost constrained. So you have to verify that these constraints are verified all along the design flow. The constraints can be used together with classical critera, as the

G. Saucier et al. (eds.), *Logic and Architecture Synthesis*

efficiency, the quality, in order to select an algorithm, or a class of algorithm, for the application.

Our approach looks to define a methodology which allows to take into account these three main points. It has three main characteristics. On the first hand, it is dedicated to signal processing application with real time and cost constraints. On the other hand, the method is based on the interaction between architectural and logical synthesis tools. Finally, in order to fit with the application field and reduce the complexity of the approach, the architectural model has been restricted to DSP. The paper will be organized around four main points: an overview of the tool, the link with the logical synthesis tool, results, and work currently in progress.

2 AN OVERVIEW OF GAUT

The VHDL compiler is integrated as a front end of the framework GAUT (Martin 1993). With regard to the implementation of procedures, the designer has two options: a hardware component or a software description. A hardware solution has the advantage of a good area-time trade-off. However, a logical solution leads to the selection of standard operators that can be re-used for other operations. This finally achieves a smaller global area of the architecture. In this case the procedures will be inlined during the synthesis. The best method depends on the use rate of the macro-function.

After dependence analysis, the front end translates the entry point into a signal flow graph representation. During this step, several transformations are realized such as procedure inlining, fixed-bound loops unrolling, variable propagation,.... This provides a unique assignment representation. Other transformations are realized on the signal flow representation, such as the transformation of conditional constructs in order to obtain a deterministic flow graph. From this internal representation, the operators are selected, the parallelism is reduced, the operations are assigned to operators, and some optimizations are carried out. Next, the description of the dedicated architecture and its controler will be produced. The architecture satisfies the real time constraint that was specified at the beginning of the synthesis.

3 HIGH LEVEL SYNTHESIS AND ITS CONNECTION WITH A LOGICAL SYNTHESIS TOOL

The design flow is represented in Figure 1. The component library contains various operators (multidelay, multifunctional, pipeline). The generic aspect of the components is computed by using the logical synthesis tool. Generic parameters give information such as area, delay, bit size, functions available (e.g. "+","-","shift", ... for an ALU), pipe stages, etc...

Architectural synthesis enables the re-use of operators. So, after the creation of a data flow graph from the VHDL specification, after transformations, the allocation-selection stage included in GAUT selects from the library the type and the number of operators in order to respect a real time constraint with a minimal cost area. It enables the use of an efficient and exhaustive research in relatively complex libraries. With a branch and bound method, the optimal solution is reached with a very short computation time. This task is realized before the time-scheduling and the assigment that minimize the cost of registers and interconnections.

After the selection of the optimal operators and the scheduling of the flow graph, GAUT produces a VHDL architecture. This description includes the behavioural description of the memory unit and the structural description of the the processing unit. The control unit is described as a flattened finite state machine with the help of behavioural VHDL. Its up now, to the logical synthesis to optimize and map out a gate level hardware structure.

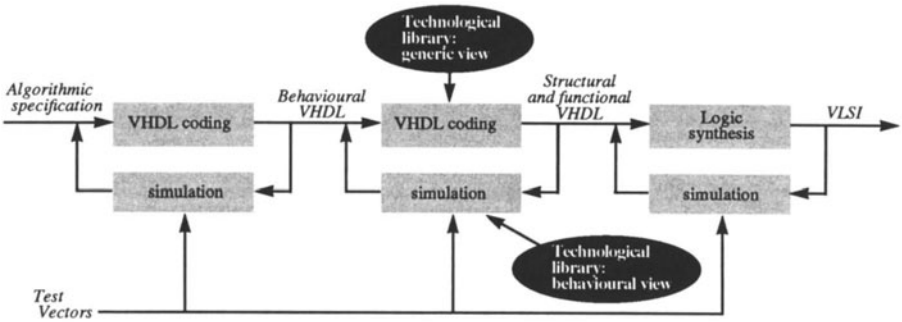

Figure 1 Design flow

The test vectors used in the simulations of the design flow depend on the application. The processing unit is organized around cells (see figure 2) : a cell may integrate many registers or share a register with other cells through the use of multiplexer/demultiplexer and tristate modules.

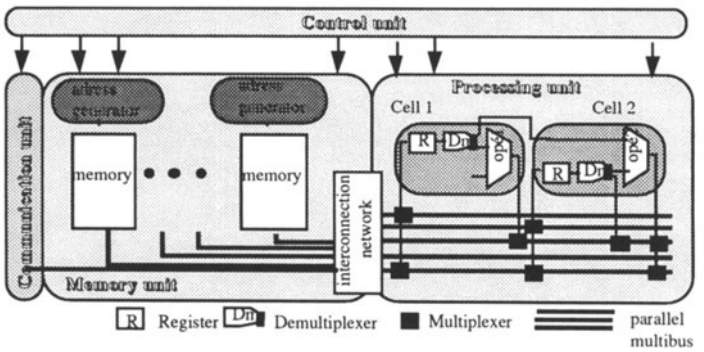

Figure 2 Generic architecture of a dedicated processor.

The control model of the architectures synthesized by GAUT integrates two features : the use of the pipeline for registers and operators, and the use of a multi-phase clock to control the events in order to optimize the efficiency of each cell . It is implemented with a fast clock representing the minimal phase of the architecture (GCD of the time of the set of operators and of the memory access time). This minimal phase can be controled by the user. The regularity of the program is used in order to reduce the time of the logical synthesis and the complexity of the routing.

Digital Signal Processing applications all have the distinctive feature of producing FSM descriptions with a lot of states but with some regularities in the control of the datapath. Reducing the number of states can be done by eliminating the No OPerations of the description or by searching the repetitive sequence of the FSM graph in order to transform them into cycles. We introduce a counter in our generic model to implement these optimizations. Let us consider the following example in which the processing time of the operator 1 is four times greater than the processing time of the operator 2.

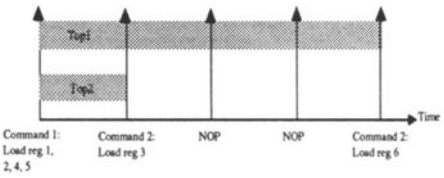

Figure 3-a Result of the synthesis: Sequencing of the operations on the data path (5 states before optimization).

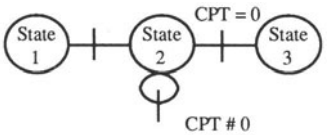

Figure 3-b Result of the synthesis: 3 states by using a counter after optimization

In figure 2 we see that the control of one cell need several signals : multiplexer, demultiplexer and tristate control, synchronous load of registers, control of pipeline and multifunction operators. Therefore, the routing area between the control unit and the processing unit may increase significantly with the complexity of the data path. Almost of DSP applications need in fact a very few set of commands. This instruction will be routed on the circuit in order to be decoded by the cell itselve to decrease the layout synthesis task and to optimize the area cost of the chip.
The control unit is organized around the generic model of the figure 4. The Moore finite state machine, the decoders and the counter are described in VHDL in order to be synthesized by logic synthesis tool (Compass, Altera, ...). The user can easily control the floorplanning of the all circuit.

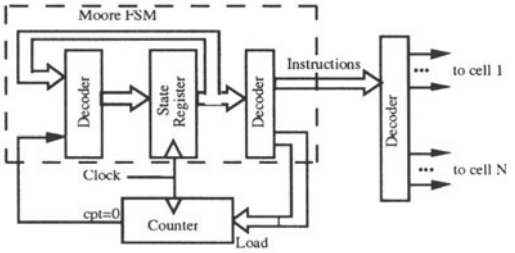

Figure 4 Generic model of the control unit.

4 RESULTS

We first describe some results based on an echo cancellation application (Gilloire 1987), a 1µ library from VLSI and the use of GAUT and COMPASS. Figure 5 gives the layout of a

LMS (1024 weights, fmax=16kHz) obtained from the cooperation of GAUT and COMPASS. The generic aspects of some library components were computed by using the Compass datapath synthesis tool (e.g. the multiplier), or the Compass VHDL synthesis tool (e.g. the adder/substracter). This architecture has been automatically produced from a behavioural description and component library (fig. 6).

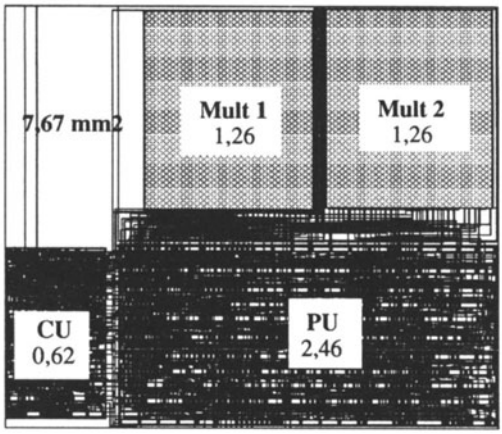

Figure 5 LMS 1024 weights, sampling frequency : 16kHz, with improved routing

BEHAVIORAL DESCRIPTION	LIBRARY DESCRIPTION

ENTITY lms IS
GENERIC (latency_time : time := 62500);
PORT (xt,yt:IN integer; yct:OUT integer);
END lms;
 	
ARCHITECTURE behavioural OF lms IS
CONSTANT N: integer := 1000;
TYPE vec is array (0 TO N) OF integer;
SIGNAL x, H : vec;
BEGIN
PROCESS
VARIABLE adapt,mu, y: INTEGER;
VARIABLE i: CONTROL;
BEGIN
y := xt * H(0);
FOR i IN 1 TO N-1 LOOP -- filtering
y:= y + x(i) * H(N-i-1);
END LOOP
yct <= y;
adapt := (yt - xt) * mu;
FOR i IN 0 TO N-1 LOOP -- adaptation
H(i) <= H(i) + adapt * x(N-i-1);
END LOOP;
FOR i IN 1 to N-2 LOOP -- signal variation
x(N-i) <= x(N-i-1);
END LOOP;
x(1) <= xt;
WAIT FOR latency_time
END PROCESS;
END behavioural;
	PACKAGE VLSI_GAUT IS
	CONSTANT nb_bit : integer:= 16;
	COMPONENT register16
	GENERIC (area : integer:= 139;
	real_func: one_function:= (reg);
	time_func: proc_time := (6);
	nb_input : integer:= 2;
	nb_pipe: integer:= 1););
	END COMPONENT;
	COMPONENT Adder_Substracter16
	GENERIC (area : integer:= 252;
	real_func: two_function:= (add,sub);
	time_func: proc_time := (15,15);
	nb_input : integer:= 2;
	nb_pipe: integer:= 1);
	END COMPONENT;
	COMPONENT multiplier16
	GENERIC (area : integer:= 2032;
	real_func : one_function := (mul);
	time_func : proc_time := 60;
	nb_input : integer:= 2;
	nb_pipe : integer:= 1);
	END COMPONENT;
	...
	END VLSI_GAUT ;

Figure 6 *Implementation of a LMS component: behavioural specification and library.*

To control the data path, among one thousand states, only twelve different commands are necessary. This is a result of the program's regularity. In order to reduce the routing between the control and processing units, a coding/decoding mechanism has been automatically introduced by GAUT. The number of states is reduced through the use of a counter. After these optimizations, the time for logic synthesis and routing has been reduced almost by half.

Others applications or technologies can be used. The figure 7 gives some results of a ADPCM algorithm used in speech coding (CCITT 1988). The results are given for two technologies and two specifications (one using multiplication operators, the other booth functions).

	ALTERA 8 bits Library	**COMPASS Library 1µm**
ADPCM 1	1 multiplier, 1 add_sub, 1 comparator, 26 registers, 5 mux, 24 demux, 1 step pipeline **area : 825 cells**	1 multiplier, 1 adder, 1 substractor 1 shifter, 1 comparator, 43 registers 14 mux, 43 demux, 1 step pipeline **area : 14,3 mm^2**
ADPCM 2: with booth functions	1 add_sub, 2 comparators, 49 registers, 1 shifter, 15 mux, 49 demux, 1 step pipeline **area : 1441 cells**	1 add_sub, 2 comparator, 1 shifter, 1 alu (+ ; - ; and), 39 registers, 14 mux , 39 demux, 1 step pipeline **area : 11,2 mm^2**

Figure 7 ADPCM algorithm synthesis; two algorithmic specifications and two libraries

5 CONCLUSION AND FUTUR PROSPECTS

The methodology which has been presented is oriented towards fast prototyping. It allows the re-use of the results of previous developments and the specification of user constraints such as real time processing. It has been tested on real applications.

The work on ADPCM (Diguet 94) underlines the importance of the style of algorithmic specification. Different studies are currently in progress on the adequacy between the algorithm and the architecture to modelize the links between algorithmic style and hardware complexity. From an analysis of an architectural synthesis result, the purpose is to introduce transformations at different levels. The transformations have to be done to improve the adequacy algorithm-architecture in respecting a real time constraint. These transformations of the specifications can occur at several levels:

Algorithmic level: A DSP application can be realized by using various algorithmic equivalent processes, for instance, the choice between a lattice or a block version of an adaptive filter.

Structural level: The techniques which we are using are for the most part known. They are based on the properties of associativity, distributivity and commutativity. In addition we are using the fusion of equivalent nodes, the elimination of redundancy and loop transformations. We are working on transformations to improve the regularity of a flow graph.

Functional level: A given operation can be mapped on different kinds of operators or sets of operators. For example, a multiplication can be executed by a hardware multiplier or by the association of a shifter, an adder and a software function. Another example: it can be profitable for a convolution to replace a multiplier and an adder by a multiplier-adder ("mad"). The interest of such modifications depends on the regularity and the rate of use of the transformed nodes among the operations of the algorithm.

Another purpose is to extend the approach in terms of constraints and classes of architecture. This is currently being done by taking into account the consumption aspect, and a model of architecture based on programmable DSP and ASICs.

5 REFERENCES

CCITT (1988) Rec.G727 Melbourne.
Gilloire A., Jullien J.-P. (1987) L'acoustique des salles dans les télécommunications. *L'écho des RECHERCHES* ,N°127.
Martin E., Sentieys O., Dubois H., Philippe J.-L. (1993) GAUT: An Architectural synthesis Tool for Dedicated Signal Processors *EURO-DAC*.
Philippe J.-L, Sentieys O., Diguet J.-P., Martin E. (1994) Adequacy architecture algorithm, an experiment in signal processing by using FGGA *VHDL forum*, Tremezzo, Italy.
Sentieys O., Martin E., Philippe J.L. (1994) VLSI architectural synthesis for an acoustic echo cancellation application *VLSI signal processing VI* page 84- 92.
Diguet J.P., Sentieys O., Philippe J.L., Martin E.; "How Specify an Algorithm in VLSI architectural Synthesis, a Vocal Coding Application"; *VLSI Signal Processing VII*; La Jolla-San Diego; October 94; pp 346-355.

6 BIOGRAPHY

Jean- Luc Philippe received the Ph.D. in signal processing from the University of Rennes in 1984. Since 1992 he is working as an assistant professor at ENSSAT, University of Rennes, conducting a research group in CAD for VLSI design. His research interests include signal and image processing systems, VLSI and high level synthesis.

Olivier Sentieys graduated from ENSSAT as a senior executive engineer in Electronics and Signal Processing Engineering in 1990, and received the Ph.D. degree in Signal Processing and Telecommunications in 1993 from the University of Rennes. Since 1990 he has been working as an Assistant Professor at ENSSAT, and is involved in research of high level synthesis of VLSI systems dedicated to signal processing , and in the development of the synthesis tool GAUT.

Eric Martin received the professor agregation at ENS de Cachan in 1984, the Ph.D at university of Orsay in 1986 and the Habilitation at University of Rennes in 1993. Since 1994 he is Professor at the UBS (Universite de Bretagne Sud), and director of the research laboratory LESTER. His research activities are in the field of high level CAD tool development for real time signal and image processing implementation.

Jean-Philippe Diguet graduated from ESEO as a senior executive engineer in Electronics and Signal Processing Engineering in 1992, and received the DEA degree in Signal Processing and Telecommunications in 1993 from the University of Rennes.Presently he is pursuing a Ph.D.degree in the LASTI at ENSSAT, and is involved in research of high level synthesis of VLSI systems dedicated to signal processing and adequacy algorithm - architecture.