Data Wordlength Optimization for FPGA Synthesis

Nicolas HERVÉ, Daniel MÉNARD and Olivier SENTIEYS

IRISA – University of Rennes
6, rue de Kerampont
22300 Lannion, France
{first-name}.{name}@irisa.fr

Abstract

Field Programmable Gate Arrays (FPGAs) are now considered as a real alternative for Digital Signal Processing (DSP) applications. But, new methodologies are still needed to automatically map a DSP application into an FPGA with respect to design constraints such as area, power consumption, execution time and time-to-market. Moreover DSP applications are frequently specified using floating-point arithmetic whereas fixed-point arithmetic should be used on FPGA. In this paper, a high-level synthesis methodology under constraints is presented. The originality is to consider a computation accuracy constraint. The methodology is based on a fixed-point operator library which characterizes the operators cost according to their wordlength. An error noise propagation model is used to compute an analytical expression of the accuracy in function of the signals wordlength.

To obtain an efficient hardware implementation, the data wordlength optimization process is coupled with the high-level synthesis . In addition, the accuracy evaluation is done through an analytical method, which drastically reduces the optimization time.

Key Words :

Fixed-point arithmetic, FPGA, high-level synthesis, CAD tools

I. Introduction

Reconfigurable hardware and especially FPGAs are more and more used for intensive digital signal processing applications such as multimedia or telecommunication. FPGAs are now able to over-perform general purpose processors thanks to their algorithm evolution adaptability. Moreover, the global performances are in constant improvement (some millions gate-equivalent, large memory, dedicated arithmetic operators, embedded microprocessors, ...). The main problem, slowing down their intensive use is the lack of high-level development tools to target theses architectures from algorithmic specification language or formalism.

Efficient implementation of DSP algorithms in embedded systems needs to use fixed-point arithmetic. In the case of fixed-point architectures, operators, buses, and memories require less area compared to their equivalent floating point arithmetic, and the associated power consumption is also lower. Furthermore, floating-point operators are more complex and lead to longer execution time, because they have to manipulate the data mantissa and exponent.

Floating-point to fixed-point conversion is a tedious and an error prone task. The hardware implementation process has to define the complete architecture. For the processing part, the operator number and type must be defined but also the wordlength of these operators. For complex designs, the wordlength search space is too large for a manual exploration. Thus, time-to-market reduction requires high-level tools to automate the fixed-point architecture synthesis.

The fixed-point conversion process must determine, for all data, a wordlength and a binary point position. The first step of this process is the data dynamic range evaluation. These results are used in the second step to determine the binary point locations for each data. The third step objective is to fix the data wordlength such that the global cost (e.g. total area) is minimized and the accuracy constraint is satisfied. Whereas the first two steps may be determined once for all, the cost minimization step must be achieved with successive refinements. Best results are obtained when the optimization process is coupled with the high-level synthesis process [1]. On the one hand, high-level synthesis requires operator wordlength knowledge to correctly execute its selection, scheduling and resource binding steps [2]. On the other hand, the optimization requires resource binding and scheduling information to correctly group data. To deal with this, an iterative refinement process must be used.

Most of the published methodologies do not realize a coupling between the data wordlength optimization and the high-level synthesis processes, and those using simulation based accuracy evaluation techniques lead to prohibitive optimization time.

In this paper, a new methodology to automatically implement a floating-point algorithm into an FPGA using fixed-point arithmetic is proposed. An iterative process on high level synthesis and data wordlength optimization is used to improve both of these dependent processes. Furthermore, the accuracy evaluation is done through an analytical approach which reduces dramatically the

optimization time.

This paper is organized as follows. Section 2 presents related works in the area of multiple wordlength architecture design. Section 3 describes our methodology design flow. Section 4 details the optimization process and section 5 shows results obtained on an application example.

II. Related Works

A. Accuracy evaluation

The most common used metric to evaluate the fixed point implementation accuracy is the Signal to Quantification Noise Ratio (SQNR), but other metrics could be used such as worst case output error [3]. Two kinds of methods can be applied to determine this metric: simulation based methods [1], [4] and analytical methods [5], [3], [6].

Simulation based methods estimate the accuracy statically from signal samples obtained after fixed-point and floating-point simulations. The floating-point result is considered as reference because the associated error is negligible compared to the fixed-point one. The fixed-point simulation requires to emulate all the fixed-point arithmetic mechanisms. As a consequence, the simulation time is considerably increased. Moreover, to obtain an accurate evaluation, a great number of samples is necessary. The combination of these two phenomena leads to long simulation time. The hardware design has to optimize the data wordlength according to the accuracy constraint. So the data wordlength search space have to be explored. Accuracy evaluation based on simulation methods leads to an iterative process with numerous simulations. For complex designs, optimization time becomes prohibitive and heuristics must be used to limit the search space.

In [6], an analytical method is proposed to compute the output quantification noise power. This method is based on a noise propagation model. Several symplifying hypothesis are made, to obtain output noise expressions independent of the input signal statistics. This model is valid only for operators using a rounding quantification law and cannot deal with cyclic graphs like in the case of recursive systems.

B. Data Wordlength optimization

The classical approach used to optimize data wordlength consists of using a unique fixed-point format for data [6]. This reduces the search space to one dimension and simplifies the synthesis because all operations will be executed on same wordlength operators. However, to obtain an efficient fixed-point DSP algorithm implementation, a specific format for each data has to be considered [5]. Some commercial solutions like Matlab fixed point toolbox, or the recent Catalytic softwares helps the user to chose adequate wordlengths, but the search for optimization is not automated and there is no support for High Level Synthesis (resource sharing, scheduling, ...). This paper focus on multiple wordlength high level synthesis (MW-HLS).

The methodology proposed by Constantinides *et all.* [5] is made up of two steps. The first one gives a fixed-point specification that respects the accuracy constraint. For this step, a dedicated resource to each operation hypothesis is used. So, the architecture synthesis is not considered first, because there is no resource sharing. The FPGA cost model corresponds to the area used in the case of a direct implementation of the Signal Flow Graph (SFG). The wordlength optimization is done with an integer linear programming (ILP) algorithm. A suboptimal solution may be obtained with a greedy heuristic. The method is purely analytical and is only valid for linear time invariant (LTI) signal processing systems. The second step of the process corresponds to architecture synthesis. In [7], a heuristic to combine scheduling and resource sharing is proposed for a SFG with different wordlengths. First, a scheduling with incomplete wordlength information is done, then resource sharing and selection are combined. This process will allocate operations to larger operators to maximize the use of available resources instead of allocating new resources. As concluded in [5] performing wordlength optimization before architectural synthesis can lead to sub-optimal designs. They invite to investigate the interdependence between these two steps.

In [1], the authors propose a method where the architecture synthesis is achieved during the wordlength optimization phase. The authors take into account the resource sharing to reduce the hardware cost but also to reduce the optimization time. Indeed, the accuracy evaluation is obtained through fixed-point simulations. So, heuristics are used to limit the search space and to obtain reasonable optimization time. A first step analyzes the application SFG and groups some data according to rules. For example, addition inputs will be specified with the same fixed-point format. The second step will determine the required minimum wordlength for each data group. Then, this fixed-point specification is scheduled and groups are binded to operators using the wordlength found in the previous step. During the combined scheduling-binding, some operations are binded to larger operators. The last step corresponds to operator wordlength optimization. The synthesis and wordlength optimization processes have to be interactive and to be finally terminated with a synthesis to exactly implement the fixed-point specification optimized for the given accuracy constraint. Indeed, the last step of the methodology proposed in [1] optimizes the operator wordlength. But this process can challenge the scheduling obtained in the previous step.

In this paper, a new architecture synthesis methodology under accuracy constraint is proposed. An iterative process is used to link architecture synthesis and wordlength optimization processes. In addition, the accuracy evaluation is achieved through an analytical approach leading to reasonable optimization time.

III. Fixed-Point Arithmetic Operator Synthesis on FPGA

For area and propagation time reasons, it is generally advantageous to use fixed-point arithmetic on an FPGA.

Architecture synthesis is build on a fixed-point arithmetic operator library dedicated to an FPGA family. Each operator is characterized by the number of resources used (logic cells, dedicated multipliers, etc.), propagation time and power consumption for different input and output wordlengths. These information are used in the architecture synthesis and wordlength optimization processes. In this paper, the characterization is obtained through the synthesis of these operators with the Synplicity Synplify Pro tool [8] for Xilinx Virtex-II FPGA components. The mean consumption power of these components is characterized with Xilinx ISE/Xpower. The information are finally saved as an XML database exploited in our methodology.

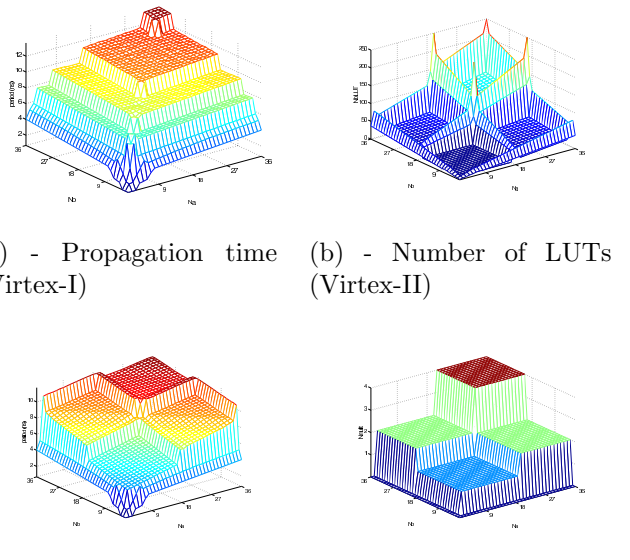
Results obtained for the multiplication operation synthesis on FPGA Virtex-I and Virtex-II are presented in Figure 1. Virtex-I does not integrate dedicated multiplier, thus this operator is implemented with 4-input Look-Up Tables (LUT). Figure 1.a shows the evolution of propagation time on the multiplier in function of the input wordlength. This curve presents stable periods showing that, in some specific points, saving one bit can reduce the propagation time of up to two nanoseconds.

Recent FPGAs, such as Virtex-II or Stratix, contain dedicated resources (multipliers, memories, data path). These resources should be integrated in the cost model. Instead of trying to uniformize these resources with only one cost metric, each specific resource is processed separately. So the user should specify a limit either for the amount of LUT or for each specific resource.

Results obtained for the Virtex-II are presented in Figures 1.b, 1.c, 1.d. This FPGA integrates 18×18 bits multipliers. In some case, the synthesis tool prefers to add LUTs to optimize the number of dedicated multiplier used. These particular cases can lead our optimization process to avoid these specific wordlengths.

IV. High-Level Synthesis with Accuracy Constraint

Our automatic implementation methodology of a floating-point algorithm into an FPGA using fixed-point arithmetic is detailed in figure 2. This approach is based on the definition of a co-synthesis and wordlength optimization environment. The proposed methodology consists of two distinct steps. First, the data binary point position is determined. This step will allow to define for each data the integer wordlength that guarantees no overflow. The data dynamic is evaluated and the results are used to define the binary-point position. Shift operations, required to obtain a correct fixed-point specification, are inserted. This step is detailed in section IV-A.



(a) - Propagation time (Virtex-I)

(b) - Number of LUTs (Virtex-II)

(c) - Propagation time (Virtex-II)

(d) - Number of 18×18 bits mutlipliers (Virtex-II)

Fig. 1. Propagation time and number of resources used in function of input wordlength for a multiplier.

Then, the multiple wordlength architecture synthesis is achieved. The objective is to minimize the resource cost as long as the accuracy constraint is verified. Thus, the process must evaluate the computation accuracy and the resource cost. These evaluations are presented in sections IV-B and IV-C.

The multiple wordlength architecture synthesis is an iterative process which is explained in section V. This process iteratively achieves, data grouping, group wordlength optimization and architecture synthesis. This last step is done with the BSS tool [9].

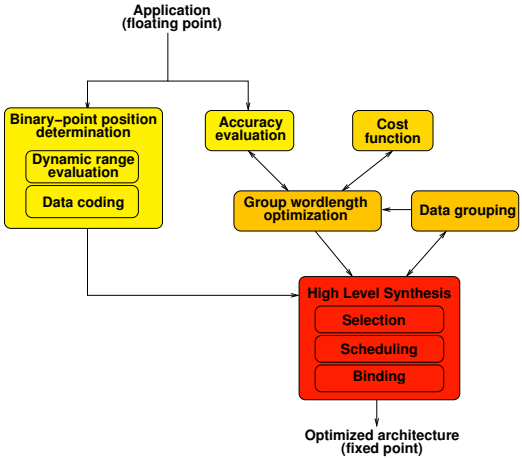


Fig. 2. Co-synthesis and wordlength optimization environment

A. Binary Point Position Determination

The aim of this step is to determine the binary point position (BPP) of each datum. This position should permit to represent the data extreme values without overflow, or with a low overflow probability and should minimize the number of bits used to code the integer part. To correctly determine this BPP, the dynamic range of each variable has first to be evaluated.

1) Data Dynamic Range Evaluation: An analytical approach has been used to guarantee that no overflow occurs. This technique determines the dynamic range expression from the system input definition domain. For non-recursive systems, interval arithmetic is used. Input data definition domain is propagated in the application data flow graph. For each operator, output data definition domain is computed from the input operands according to operator specific rules. These rules give the definition domain in the worst case. For linear time-invariant systems the dynamic range is determined through the L1 norm [10]. This norm is derived from the transfer function between the target data and the inputs. This transfer function is obtained from the application Data Flow Graph (DFG) [11]. Thus, the use of transfer function works either with recursive or non-recursive structures.

2) Data Coding: The previous step has determined the data dynamic range. These results are used to determine the minimal number of bits required to code the integer part. The aim of this step is to obtain a fixed-point specification without overflow. The different rules of the fixed-point arithmetic must be respected. Consequently, shift operations are inserted to adapt data formats to their dynamic range or to adapt the input adder format. To determine data format, propagation rules are defined for each operator type. Then, the propagation rules are applied to each operator during the application DFG parsing.

B. Accuracy Evaluation

An analytic method to evaluate precision is used [11]. As the method is analytic, optimization times are reasonable. This approach determines automatically analytical expressions of the output quantification noise power and the output signal noise power for LTI systems and non-recursive non-linear systems.

The output quantification noise is modeled through a weighted sum of noise sources. The different sources are made up of noises associated to inputs and noises generated during data wordlength reductions.

Statistical parameters associated to a noise source are derived from the data format after quantification and from the number of bits eliminated. Gain between the output and each noise source is determined depending on the type of system. For LTI systems, it is determined from the transfer function between input and output. The technique used to automatically determine the system transfer functions and the developed tool used for this technique is described in [11]. For non-linear and non-recursive systems the gain is obtained from statistical parameters of the different signals located between the output and the noise source [12]. These statistical parameters are determined with a unique floating-point simulation. From this, the SQNR expression in function of the fixed-point data formats can be obtained.

C. Cost Model

The hypothesis that the cost of various operators on an FPGA is equal to the sum of the cost of each component will be used in the cost function of the proposed methodology. By default our cost model is an approximation of the number of LUTs used by the functional units on the FPGA. A function gives the number of LUTs consumed according to the input wordlength. These information are stored in a database storing the characterization synthesis results. For FPGA integrating dedicated resources, the user has to define a maximum for each resource type. It will be easy to modify later the cost function to take into account, for example, the energy consumption. The user will have to fix weight to indicate the relative importance of consumption and resource usage in the optimization process.

V. Multiple Wordlength Architecture Optimization

Our architecture design methodology is based on an iterative process for synthesis and wordlength minimization. The aim is to find the optimal allocation which minimizes the cost through wordlength minimization. The method combines efficiently resource sharing, obtained through architecture synthesis, and wordlength optimization. This process is made-up of four steps.

A *group* is defined as a set of operations that will be computed on a same wordlength operator. The first step of the process defines the number of groups needed for each type of arithmetic operation (see V-A). In the second step, a grouping algorithm is apply. The technique used is presented in subsection V-B. The third step searches the optimal wordlength combination for this grouping. The objective is to minimize the cost under an accuracy constraint $RSBQ_{min}$ as describe in the following equation:

$$\min_{\forall \overrightarrow{wl}} \left(\text{Cost}(\overrightarrow{wl}) \right) \quad \text{with} \quad SQNR\left(\overrightarrow{wl}\right) \geq SQNR_{min}$$

where $\overrightarrow{wl}$ denotes a set of wordlengths fixed for each data in the DFG. The accuracy evaluation is done with the analytical method presented in IV-B. The cost is evaluated with the cost function presented in section IV-C. The overall optimization algorithm is presented in section V-B.

The fourth step of the method is the architecture synthesis of the fixed point specification obtained in the third step. After this synthesis, the number of operators used for each operation type has to be reconsidered. Indeed, operation wordlengths have been reduced leading to the decrease

of the operator latency. This can offer the opportunity to reduce the number of operators during the scheduling. Thus, an iterative process is necessary to converge to an optimized solution. This algorithm stops when successive iterations lead to the same results.

A. Data grouping

Data grouping is done from synthesis result analysis. A group is defined for each operator and for each operation a *mobility* is computed. The *mobility* is define as the difference between the execution date obtained for two list scheduling : one in the direct sense and the other in the reverse sense. Operations are treated by decreasing order of priority. The mobility is used to allocate operations to the most adequate operator.

To group the operations, the optimized wordlength associated with each operation is considered. This *specific wordlength* is obtained by using the spatial implementation, i.e. where each operation has a dedicated fixed-point operator.

An operation is preferably placed on the group with the wordlength immediately superior to the *specific wordlength* of that operation. The idea of this approach is to obtain for each operation the smaller wordlength and to favor placement in smaller wordlength groups.

For the first optimization step, hardware synthesis is done with a single group for each operation type. The wordlength of a given group is set to the maximum wordlength of the operations in this group. For next optimization steps, some operations will be placed on larger-than-needed operators to reduce the number of operators.

B. Group Wordlength Optimization

Optimization algorithms used are independent procedures, which for a given data grouping, will search a wordlength combination minimizing the cost function under an accuracy constraint. For each tested combination, the SQNR evaluation function and the cost function are evaluated. Their results will determine the next step of the optimization algorithm.

An optimization procedure of the group wordlengths is described here. In the first step the *minimum group wordlength set* is determined. For that, all group wordlengths are initially set to a maximum value. So the precision constraint must be satisfied. Then, for each group, the minimum wordlength still satisfying the precision is determined, all other group wordlengths staying to their maximum value. Each group is then set to its minimum. In this combination the accuracy constraint will surely not be satisfied anymore. But the advantage of this starting point is that wordlengths only have to be increased to get the optimal wordlength combination.

Next steps will consist of reaching the accuracy constraint, limiting in the same time the cost increase. The ratios $\Delta_{\mathrm{SQNR}}/\Delta_{\mathrm{cost}}$ are computed for every 1-bit wordlength increase on a single group. The group for which the wordlength increase gives the highest ratio has its wordlength indeed incremented.

VI. Implementation Results

The application chosen to underline the interest of a multiple wordlength optimization technique under an accuracy constraint is an infinite impulse response (IIR) filter. This filter is an eighth-order IIR filter implemented as four cascaded second-order cells. The signal flow graph (SFG) of this IIR filter is presented on figure 3. It is composed of 20 multiplications and 16 additions.

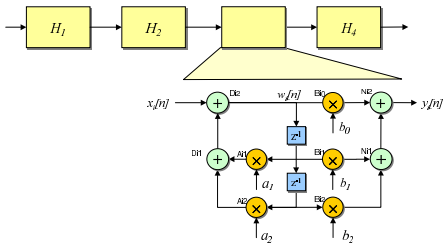
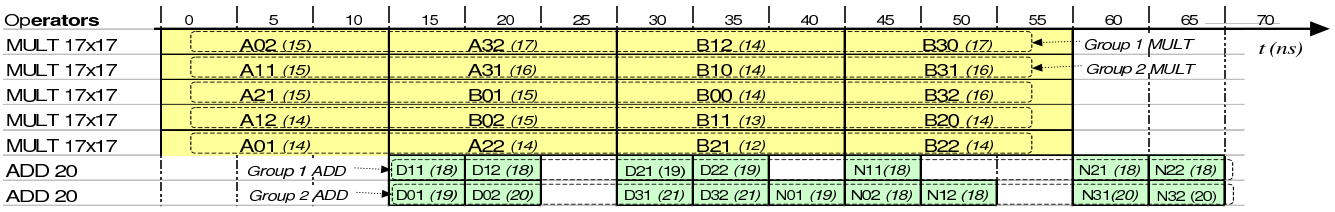


Fig. 3. Signal Flow Graph of the 8th-order IIR Filter

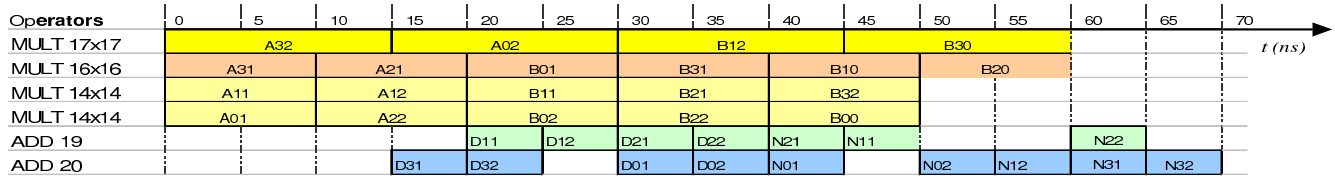
The method presented in section IV has been used to obtain the data dynamic and the binary point position and thus, a correct fixed point specification. The SQNR analytical expression has been determined. For the multiple wordlength optimization, the accuracy constraint has been set to 60 dB.

First, the different operation wordlengths have been optimized for a spatial implementation. In this case an operator is dedicated to each operation. The wordlengths obtained are presented in figure 4.a. For the first global optimization step, a group is defined for each operator type and its wordlength is fixed to the maximal wordlength value in the group. Thus, multiplications are executed on a 17×17-bit multiplier and additions on a 20-bit adder. The hardware synthesis for this fixed-point specification leads to the scheduling presented in figure 4.a. The minimal system clock frequency is set to 200 MHz, so the latency of each operator is a multiple of 5 ns. For a 70 ns time constraint, five multipliers and two adders are needed. In next step, five new groups for multiplications and two new groups for the additions are defined. These groups, presented in figure 4.a, are built depending on operation *specific wordlength* and mobility.

A group wordlength optimization, under the accuracy constraint, is done for these seven groups. This optimization gives lower wordlengths. The wordlength of the five multiplication groups are respectively 17, 16, 15, 14 and 14 bits (only operations with two equal operands wordlength have been considered). The hardware synthesis for this new fixed point architecture leads to the scheduling presented in figure 4.b. Given that, below 16



(a)



(b)

Fig. 4. Schedulings obtained for two steps of the multiple-wordlength-architecture optimization.

bits, multipliers have a latency lower than 10 ns, only four multipliers are now needed. Thus, this architecture uses one multiplier less than the previous one. The wordlength reduction combined with the operator number decreasing, reduce the area by 35%.

With the method propose in [6], a uniform wordlength architecture optimization would have lead to five multipliers and two adders with 19 bits precision. Compared to this architecture, the total area saved on operators is 47%. Theses results show the interest of using multiple wordlength architecture and our approach efficiency.

VII. Conclusion

In this paper, a new methodology for high-level synthesis with accuracy constraint has been presented. An iterative process on hardware synthesis and group wordlength optimization is used. Moreover, the accuracy evaluation is done through an analytical approach to obtain reasonable optimization time in comparison with simulation based methodologies. A tool to automate this methodology is currently under development. First results show the efficiency of our methodology and the interest of this approach for architecture area optimization compared to other methods.

References

[1] K. Kum and W. Sung, "Combined Word-Length Optimization and High-level Synthesis of Digital Signal Processing Systems," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, Aug. 2001, pp. 921–930.

[2] D. Gajski, N. Dutt, A. Wu, and S. Lin, *High-Level Synthesis*. Kluwer Academic, 1993.

[3] S. Wadekar and A. Parker, "Accuracy Sensitive Word-Length Selection for Algorithm Optimization," in *IEEE/ACM International Conference on Computer Design (ICCAD'98)*, San Jose, CA , USA, Nov. 1998, pp. 54–61.

[4] R. Cmar, L. Rijnders, P. Schaumont, and I. Bolsens, "A Methodology and Design Environment for DSP ASIC Fixed Point Refinement," in *Proceedings of the Design Automation and Test in Europe Conference (DATE 99)*, Munich, 1999, pp. 271–276.

[5] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, *Synthesis and Optimization of DSP Algorithms*. Kluwer Academic, 2004.

[6] J. Toureilles, C. Nouet, and E. Martin, "A Study on Discrete wavelet transform implementation for a high level synthesis tool," in *EUSIPCO'98*, Rhodes, Greece, Septembre 1998.

[7] G. Constantinides, P. Cheung, and W. Luk, "Heuristic Datapath Allocation for Multiple Wordlength Systems," in *Design Automation and Test in Europe (DATE 01)*, Munich, Germany, Mar. 2001, pp. 791–796.

[8] *Synplify Pro 7.3 Reference Manual*, Synplicity corporation, June 2003.

[9] O. Sentieys, J.-P. Diguet, and J. Philippe., "GAUT: a High Level Synthesis Tool dedicated to real time signal processing application," in *University Booth, EURO-DAC*, Brighton, 18-22 september 1995.

[10] T. Parks and C. Burrus, *Digital Filter Design*. Jhon Willey and Sons Inc, 1987.

[11] D. Menard and O. Sentieys, "Automatic Evaluation of the Accuracy of Fixed-point Algorithms," in *Design, Automation and Test in Europe 2002 (DATE-02)*, Paris, France, March 2002, pp. 529–535.

[12] D. Menard, R. Rocher, P. Scalart, and O. Sentieys, "SQNR determination in non-linear and non-recursive fixed-point systems," in *XII European Signal Processing Conference (EUSIPCO 2004)*, Vienna, Austria, Sept. 2004, pp. 1349–1352.