Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 906350, 14 pages
doi:10.1155/2012/906350

Research Article

High-Level Synthesis under Fixed-Point Accuracy Constraint

Daniel Menard, Nicolas Herve, Olivier Sentieys, and Hai-Nam Nguyen

IRISA/ENSSAT, University of Rennes, 6 rue de Kerampont, 22305 Lannion, France

Correspondence should be addressed to Daniel Menard, daniel.menard@irisa.fr

Received 18 July 2011; Revised 12 December 2011; Accepted 4 January 2012

Academic Editor: Philippe Coussy

Implementing signal processing applications in embedded systems generally requires the use of fixed-point arithmetic. The main problem slowing down the hardware implementation flow is the lack of high-level development tools to target these architectures from algorithmic specification language using floating-point data types. In this paper, a new method to automatically implement a floating-point algorithm into an FPGA or an ASIC using fixed-point arithmetic is proposed. An iterative process on high-level synthesis and data word-length optimization is used to improve both of these dependent processes. Indeed, high-level synthesis requires operator word-length knowledge to correctly execute its allocation, scheduling, and resource binding steps. Moreover, the word-length optimization requires resource binding and scheduling information to correctly group operations. To dramatically reduce the optimization time compared to fixed-point simulation-based methods, the accuracy evaluation is done through an analytical method. Different experiments on signal processing algorithms are presented to show the efficiency of the proposed method. Compared to classical methods, the average architecture area reduction is between 10% and 28%.

1. Introduction

Implementing signal processing applications in embedded systems generally requires the use of fixed-point arithmetic [1, 2]. In the case of fixed-point architectures, operators, buses, and memories need less area and consume less power compared to their equivalent using floating-point arithmetic. Furthermore, floating-point operators are more complex and lead to longer execution time.

However, the main problem slowing down the hardware implementation flow is the lack of high-level development tools to target these architectures from algorithmic specification language using floating-point data types. In this design process, mainly two kinds of high-level Computer Aided Design (CAD) tools are required for reducing the time-to-market: floating-point to fixed-point conversion and High-Level Synthesis (HLS).

For hardware implementation like FPGA or ASIC, the floating-point to fixed-point conversion is a complex and an error prone task that converts an application specified with high-precision floating-point data into an algorithm using fixed-point arithmetic, usually under an accuracy constraint. Then, HLS automatically translates the algorithm specified with fixed-point data into an optimized dedicated architecture. In the processing part of this architecture, the number and the type of operators must be defined Moreover, each operator input and output word-length must be determined. For complex designs, the word-length 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 process.

The aim of HLS, handling multiple word-length, is to minimize the implementation cost for a given fixed-point accuracy constraint. This process leads to an architecture where each operator word-length has been optimized. Best results are obtained when the word-length optimization (WLO) process is coupled with the HLS process [3, 4]. HLS requires operator word-length to correctly execute its allocation, scheduling, and resource binding steps. But the word-length optimization process requires operation-to-operator binding. To deal with this optimization issue, an iterative refinement process should be used. Many published methodologies [5–9] do not couple data WLO and HLS processes. Moreover, simulation-based accuracy evaluation is used, which leads to prohibitive optimization time.

In this paper, a new method for HLS under accuracy constraint is proposed. The WLO process and the HLS

are combined through an iterative method. Moreover, an efficient WLO technique based on tabu search algorithm is proposed to obtain solutions having better quality. Compared to existing methods, the HLS synthesis process is not modified and thus this method can take advantage of existing academic and commercial tools. Furthermore, the proposed method benefits from an analytical accuracy evaluation tool [10], which allows obtaining reasonable optimization times. Experiments show that good solution can be obtained with a few iterations.

This paper is organized as follows. In Section 2, related work in the area of multiple word-length architecture design is summarized. Then, the proposed fixed-point conversion method for hardware implementation is presented in Section 3. The multiple word-length architecture optimization is detailed in Section 4. In Section 5, different experiments on various signal processing algorithms are presented to show the efficiency of the proposed method. Finally, Section 6 draws conclusions.

2. Related Works

The classical method used to optimize data word-length relies on handling uniform word-lengths (UWL) for all data that reduces the search space to one dimension and simplifies the synthesis because all operations will be executed on operators with the same word-length. However, considering a specific fixed-point format for each data leads to an implementation with a reduced power, a smaller area, and a smaller execution time [9].

In the sequential method [5–9], the word-lengths are first optimized and then the architecture is synthesized. The first step gives a fixed-point specification that respects the accuracy constraint. For this purpose, a dedicated resource is used for each operation. So, the HLS is not considered first, because there is no resource sharing. The second step of the process corresponds to HLS. In [9], a heuristic to combine scheduling and resource sharing is proposed for a data flow graph with different word-lengths. This method implements a fixed-point application, which leads to a numerical accuracy greater than the constraint. In the first step, the data WLO gives a numerical accuracy close to the accuracy constraint, but, in the second step, the binding to larger operators will improve the global numerical accuracy. Consequently, the obtained solution may not be optimized exactly for the specified accuracy constraint given that the two steps are not coupled.

A method combining word-length optimization and HLS has been proposed in [11]. This method is based on a Mixed Integer Linear Programming (MILP). This MILP formulation leads to an optimal solution. Nevertheless, some simplifications have been introduced to limit the number of variables. This method is restricted to linear time-invariant systems, and the operator latency is restricted to one cycle. Moreover, the execution time to solve the MILP problems can become extremely long and several hours could be needed for a classical IIR filter.

In [3], the authors propose a method where the HLS is achieved during the WLO phase. The authors take account of 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. Therefore, 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 determines the required minimum word-length (MWL) for each data group. The MWL of a group corresponds to the smallest word-length for the data of the group allowing fulfilling the accuracy constraint when the quantization effect of the other groups is not considered. This MWL is used as a starting point because its computation can be achieved in a reasonable execution time when simulation-based methods are used to evaluate the accuracy. In the third step, the fixed-point specification is scheduled and groups are bound to operators using the word-length found in the previous step. During the combined scheduling-binding, some operations are bound to larger operators. Finally, the last step corresponds to the operator WLO.

The synthesis and WLO processes have to be interactive and have 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 method proposed in [3] optimizes the operator word-length. But this process can challenge the scheduling obtained in the previous step.

In [4], a method combining WLO and HLS through an optimization process based on simulated annealing is proposed. In the following, a movement refers to a modification in the system state of the simulated annealing optimization heuristic. This method starts with the solution obtained with uniform word-length (UWL). In this optimization process based on simulated annealing, movements on the HLS are carried-out by changing the mapping of the operations to the operators. An operation can be mapped to a nonmapped or to another already mapped resource or operations can be swapped. Movements on WLO are carried out by modifying the operation word-length. The movement can increase or decrease the word-length of a signal of one bit or make more uniform the word-length of the operation mapped to a same operator. A movement is accepted if the implementation cost is improved compared to the previous solutions and the accuracy constraint fulfilled. If the accuracy constraint is fulfilled but the implementation cost is not improved, the movement is accepted with a certain probability decreasing with time. Thus, for each movement, the implementation cost, the fixed-point accuracy, and the total latency of the current solution must be computed. Stochastic algorithms lead to good quality solutions for optimization problems with local minimum. But they are known to require a great number of iterations to obtain the optimized solution. Given that each iteration requires an architecture synthesis and an accuracy evaluation, the global optimization time can be very high.

In this paper, a new HLS method under accuracy constraint is proposed. An iterative process is used to link HLS and WLO processes, and good results are obtained

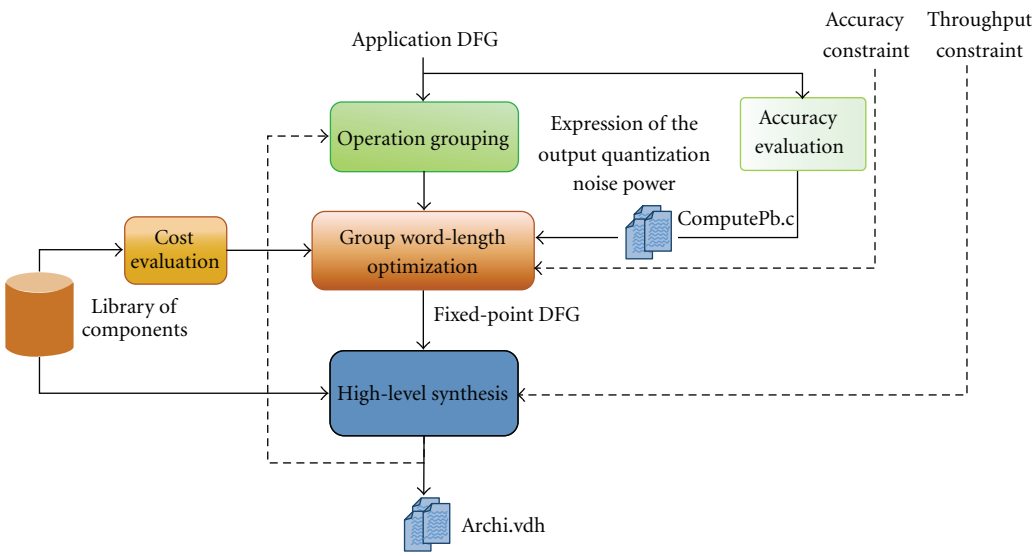

Figure 1: Combined HLS and WLO environment.

with a few iterations. The accuracy evaluation is carried-out through an analytical method leading to reasonable optimization time. Compared to [3, 4], a classical HLS tool can be used and no modification of this tool is required. Thanks to the analytical method, the optimized word-length (OWL) associated to each operation can be computed in a reasonable time and is used as starting point, as opposed to the MWL like in [3]. It is obvious that the MWL is less relevant than the OWL since that the quantization effects of the other operations are not taken into account.

3. High-Level Synthesis under Accuracy Constraint

A fixed-point data is composed of two parts corresponding to the integer part and the fractional part. The number of bits for each part is fixed. iwl is the number of bits for the integer part integrating the sign bit, fwl is the number of bits for the fractional part, while $w = iwl + fwl$ is the total number of bits. The scaling factor associated with the data does not evolve according to the data value as in floating-point arithmetic. So, the aim of fixed-point conversion process is to determine the optimized number of bits for the integer part and the fractional part.

The proposed HLS method under accuracy constraint is detailed in Figure 1. The aim is to implement an algorithm specified with floating-point data types into an architecture using fixed-point arithmetic. This method is based on the definition of a co-synthesis and WLO environment. The input of the framework, for multiple word-length high-level synthesis, is the Data Flow Graph (DFG) of the application. Nevertheless, Control Data Flow Graph can be used if the HLS tool supports this intermediate representation. For each operation operand, the binary point-position has been determined. The binary point-position must allow the representation of the data extreme values without overflow and the minimization of the number of bits used for the integer part.

In our multiple word-length high-level synthesis approach, the operator WLO and the HLS are coupled. The goal is to minimize the architecture cost as long as the accuracy constraint is verified. The multiple word-length HLS is an iterative process as explained in Section 3.1. The HLS, under throughput constraint, is carried out with the tool GAUT [12]. The HLS and the WLO process usees a library composed of various types of operator characterized in terms of performances for different operand word-lengths.

3.1. Multiple Word-Length Architecture Optimization. To obtain an optimized multiple word-length architecture, the operator word-length determination and the HLS must be coupled. Indeed, for the HLS, the operation word-length must be known. The operator propagation time depends on the input and output word-length. For the operator WLO, the resource sharing must be taken into account. A group g_i is defined as a set $\mathcal{S}_{g_i}$ of operations o_k that will be computed on the same operator α_i. To determine a group, the operation assignment must be known. The operations executed by the same operator α_i must have the same word-length w_i, and this condition must be taken into account during the optimization of the group word-length.

To couple HLS and WLO, the proposed method is based on an iterative process with the aim of finding the optimized operation binding, which minimizes the cost through word-length minimization. The method efficiently combines resource sharing obtained through HLS and WLO search. This process is presented in Figure 2. This process is in four steps.

The first step defines the number of groups needed for each type of arithmetic operation. For the first iteration, the number of group for each operation type is set to one. Indeed, the number of operators required to execute

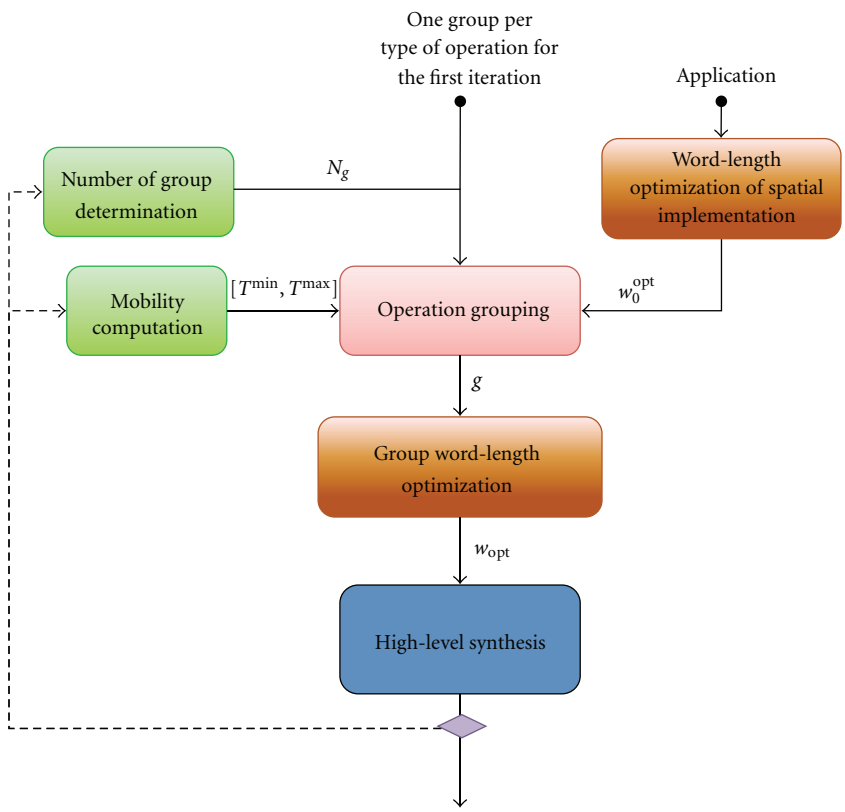

Figure 2: Iterative process for combining word-length optimization and high level synthesis.

each operation type is unknown and can be defined only after an architecture synthesis. For the other iterations, the number of groups for each operation type is defined from the HLS results obtained in the previous iteration. The group number for an operation type is fixed to the number of operators used for this operation type. In the second step, a grouping algorithm is applied. This step, relatively similar to clustering [6, 9], aims to find the best operation combinations, which would lead to interesting results for the WLO process and HLS. The technique is presented in Section 3.1.1. The third step searches the optimal word-length combination for this grouping, that minimizes the implementation cost and fulfills the accuracy constraint. This optimization process is detailed in Section 4. The fourth step is the architecture processing part synthesis from 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 word-lengths have been reduced leading to operator latency decrease. 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, and the algorithm stops when successive iterations lead to the same results or when the maximal number of iterations is reached.

3.1.1. Operation Grouping. Operation grouping is achieved from an analysis of the synthesis result. A group g_i is defined for each operator α_i of the synthesized architecture. Each group g_i is associated with a word-length w_{g_i} that corresponds to the maximal word-length of the operations o_k associated to the group g_i

$$w_{g_i} = \max_{o_k \in \delta_{g_i}} \left(w_{o_k}^{\text{opt}} \right), \tag{1}$$

where $w_{o_k}^{\text{opt}}$ is the optimized word-length associated with each operation o_k. $w_{o_k}^{\text{opt}}$ corresponds to optimized word-length obtained with a spatial implementation, that is, where each operation has a dedicated fixed-point operator. This word-length $w_{o_k}^{\text{opt}}$ is obtained with the optimization algorithm presented in Section 4.3, when a group g_k is assigned at each operation o_k.

For each operation a mobility interval $[T_i^{\min}; T_i^{\max}]$ is computed. The mobility index m_{α_i} is defined as the difference between the execution dates, $T_i^{\min}$ and $T_i^{\max}$, obtained for two list schedulings in the direct and reverse directions. Operations are treated with a priority to least mobility operations. The mobility index m_{α_i} is used to select the most appropriate group for operation o_i.

To group the operations, the optimized word-length $w_{o_k}^{\text{opt}}$ associated with each operation o_k is considered. An operation o_k is preferably associated to the group g_i with the word-length w_{g_i} immediately greater than to the optimized operation word-length $w_{o_k}^{\text{opt}}$ and compatible in terms of mobility interval. In case of mobility inconsistency, the grouping algorithm tries, firstly, to make the operation o_k take the place of one or more operations o_j having a smaller word-length $w_{o_j}^{\text{opt}}$, secondly, to place o_k in another group

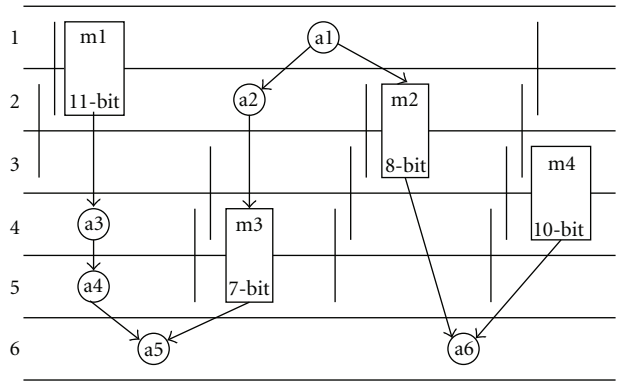

Figure 3: Application DFG showing alternative scheduling possibilities meeting timing constraints for the multiplications (no resource constraint).

Table 1: Priority list of multiplication operations with mobility index and spatio-optimal word-length.

Operation	Mobility index	$w_{o_k}^{\text{opt}}$	Priority index
m1	3	11-bit	1 (highest)
m2	3	8-bit	2
m3	4	7-bit	3
m4	5	10-bit	4

having a greater word-length and finally creates a new group with this operation o_i if other alternatives have failed. The idea of this method is to obtain for each operation the smaller word-length, and to favor placement in smaller word-length groups. When an operation o_j has been removed from a group, this operation returns to the top of the priority list of operations to be assigned. The convergence of the algorithm is ensured by the fact that operations are placed one by one in groups according to their priority and can be returned in the priority list only by operations having strictly a higher word-length.

3.1.2. Illustrative Example. The following example illustrate, the concepts for operation grouping described above. The DFG presented in Figure 3 is considered. The circles represent addition operations (named a1,...,a7), rectangles represent multiplication operations (named m1,...,m4), arrows represent data dependency, number in italics are the optimized word-length of corresponding operation, and vertical bars represent scheduling alternatives for given multiplication. The time constraint is set to 6 clock cycles. Table 1 gives the optimized word-length $w_{o_k}^{\text{opt}}$, the initial mobility index, and the associated priority for multiplication operations.

In the rest of the example, scheduling and cost of additions are not considered to simplify illustration. The latency of the multiplications is equal to two clock cycles. For the second iteration of the iterative process, the algorithm proceeds as follows.

Step 1. The ready operation m1 with highest priority is scheduled and assigned to a resource. As there is no resource selected yet, this operation defines the first resource for type m. This resource is named **M1** (with a word-length of 11 bits).

Step 2. The ready operation m2 (with highest priority now) is scheduled and assigned to a resource. As there is scheduled time on resource **M1** compatible with m2, m2 is assigned on **M1**.

Step 3. The ready operation m3 (with highest priority now) is scheduled and assigned to a resource. As there is no compatible time on **M1** regarding the mobility of m3, a second resource **M2** is created with a word-length of 7 bits.

Step 4. Operation m4 was always ready but with a lowest priority due to its highest mobility compared to the other operations. Standard list scheduling algorithm would have allocate this operation on resource **M2** since there is no more place on resource **M1**, increasing word-length of group **M2** to 10 bits. The proposed algorithm allows operation m4 to deallocate operation m2 that have smaller optimized word-length and m4 is scheduled on resource **M1**.

Step 5. This step try to realocate operation m2 on resource of immediately superior word-length, corresponding to resource **M1**. As there is no place and no operation with smaller optimized word-length, operation m2 is placed on resource **M2** and **M2** word-length is updated to 8 bits. Observe that mobility of present operation m3 is used to maximize use of resources and let operations m2 and m3 fit on resource **M2**.

After Step 5, there is no more operation to schedule and allocate, so the algorithm finished. Resource **M1** will execute operations m1 and m4 with an effective word-length of 11 bits, and resource **M2** will execute operations m2 and m3 with an effective word-length of 8 bits resulting in a smaller architecture, while a more naive algorithm would have required an 11-bit and 10-bit multiplier.

Figure 4 presents the various steps of assignments. The step number of operation assignment is indicated by circled numbers. Figure 4(a) presents Steps 1 to 3 and Figure 4(b) Steps 4 to 5 after reassignment of operation m2.

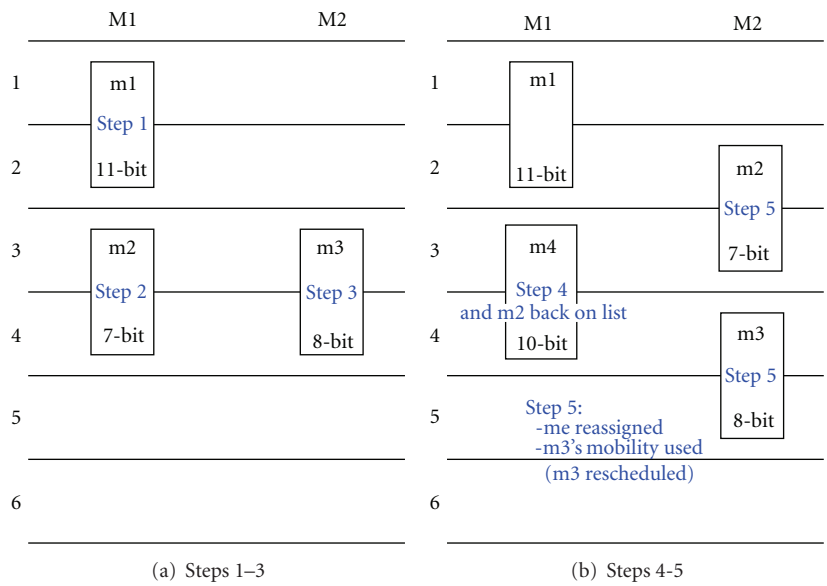

Figure 4: Scheduling with resource assignments with step information.

3.2. High-Level Synthesis Process. The high-level synthesis tool Gaut [12] is used to generate an optimized architecture from the DFG of the application. The aim is to minimize the architecture area for a given throughput constraint. The high-level synthesis process is composed of different steps. The selection module selects the best operator for each operation from the library. This library is the same as the one used for word-length optimization. Each component is characterized in terms of area, latency, cadence, and energy consumption. A list scheduling is used to schedule the operations. The algorithm is based on a mobility heuristic depending on the availability of allocated operators. Operation assignment to operators is carried out simultaneous to the scheduling task. Finally, the architecture is globally optimized to obtain a good trade-off between the storage elements (register, memory) and the interconnection elements (multiplexer, demultiplexer, tristates, and buses). Different algorithms can be used. The best results for complex applications are obtained with a variant of the left-edge algorithm.

4. Word-Length Optimization

The aim of the WLO is to find the best group word-lengths, which minimize the architecture cost $\mathcal{C}$ as long as the accuracy constraint $\lambda_{\min}$ is fulfilled. This optimization problem is described with the following expression:

$$\min(\mathcal{C}(\mathbf{w})) \text{ such as } \lambda(\mathbf{w}) \geq \lambda_{\min} \tag{2}$$

with $\mathbf{w} = [w_{g_0}, w_{g_1}, \ldots, w_{g_{N_g-1}}]$, the vector containing the word-length associated to each group. $\lambda(\mathbf{w})$ corresponds to the numerical accuracy obtained for a given group word-length $\mathbf{w}$. The evaluation of the numerical accuracy is summarized in Section 4.2.1. The cost function $\mathcal{C}$ is evaluated with the method presented in Section 4.1. For each tested combination, the accuracy and the cost function are evaluated with mathematical expressions, so that, the optimization time will be significantly reduced compared to a simulation-based method.

4.1. Cost Function. The aim of the HLS is to obtain an optimized architecture from the application functional specification. The architecture processing part is built by assembling different logic entities corresponding to arithmetic operators, multiplexers, and registers. These elements come from a library associated to the targeted FPGA or ASIC technology.

4.1.1. Generation of Operator Library. In the case of multiple word-length synthesis, the arithmetic operator library contains operators with different input and output word-lengths. The library generation flow is described in Figure 5. First, the different library elements are synthesized to obtain placed-and-routed blocks. Then, these elements are characterized by the information collected after operation synthesis.

A parameterized VHDL description is written for each operator type. From this description, a logic synthesis is achieved separately for each library element. The script-based method is used to automatically generate the library elements for different word-lengths. This logic synthesis is achieved with the Synplify Pro tool (Synopsys) for FPGA and with Design Compiler (Synopsys) for ASIC.

Let E_{Lib} denote the set containing all the library elements. Each operator α_j is characterized by the number of resources used n_{α_j} (logic cells and dedicated multipliers, for FPGA, and standard cells and flip-flops for ASIC), the propagation time t_{α_j}, and the energy consumption e_{α_j} for different input and output word-lengths. For the HLS,

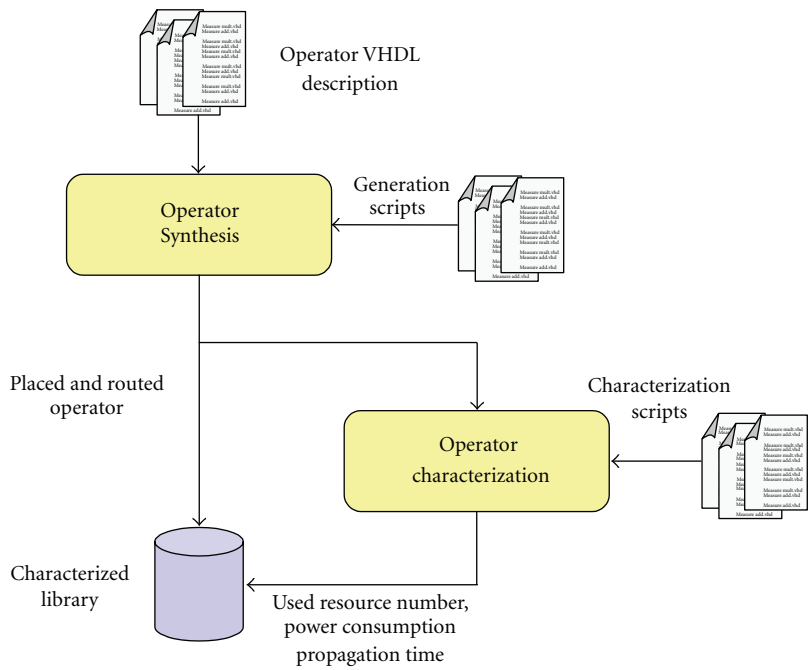

Figure 5: Operator synthesis and characterization flow.

the latency l_{α_j} of the operator is expressed in number of cycles

$$l_{\alpha_j} = \left\lceil \frac{t_{\alpha_j}}{t_{\mathrm{CLK}}} \right\rceil, \tag{3}$$

where t_{CLK} is the clock period used for the system.

This different information is used in the HLS and WLO processes. The mean power consumption of these components is characterized at the gate level with several random input vectors. The number of vectors is chosen to ensure the convergence to the mean value. This characterization is finally saved as an XML database exploited in the proposed method.

4.1.2. Model of Cost Function. The aim of the WLO is to minimize the architecture cost $\mathcal{C}$. Let S_{Opr} denote the subset of operators used for the architecture from the library ($S_{\mathrm{Opr}} \subset S_{\mathrm{Lib}}$). Let c_i denote the cost associated with each operator α_i. This cost depends on $\mathbf{w}(i)$, the word-length of operator α_i. The global cost for the architecture processing part is defined as the sum of the different costs of the operator o_i used for the architecture

$$\mathcal{C}(\mathbf{w}) = \sum_{i \in S_{\mathrm{Opr}}} c_i(\mathbf{w}(i)). \tag{4}$$

The cost used in the proposed method corresponds to the architecture area evaluated through the number of LUTs of a functional unit. For FPGA integrating dedicated resources, the user has to define a maximum number for each dedicated resource type. Moreover, other cost functions can be used to optimize energy consumption.

4.2. Constraint Function. The constraint function of the optimization problem corresponds to the fixed-point numerical accuracy. The use of fixed-point arithmetic leads to unavoidable error between the results in finite precision and in infinite precision. The fixed-point implementation is correct only if the application quality criteria are still fulfilled. Given that the link between the fixed-point operator word-lengths and the application quality criteria is not direct, an intermediate metric λ is used to define the fixed-point accuracy. The most commonly used metric is the Signal to Quantization Noise Ratio (SQNR) [3]. This metric corresponds to the ratio between the signal power and the quantization noise power.

The accuracy constraint $\lambda_{\min}$, corresponding to the minimal value of the SQNR, is determined from the system performance constraints. This accuracy constraint is defined such as the system quality criteria will be still verified after the fixed-point conversion process.

4.2.1. Fixed-Point Accuracy Evaluation. Two kinds of method can be used to determine the fixed-point accuracy. These methods are either based on fixed-point simulations or analytical. Simulation-based methods estimate the quantization noise power statistically from signal samples obtained after fixed-point and floating-point simulations [3]. The floating-point result is considered as the 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. Moreover, to obtain an accurate evaluation, an important number of samples is necessary. The combination of these two phenomena leads to an important simulation time. In the WLO process, the fixed-point accuracy is evaluated

at each iteration. For complex systems, where the number of iterations is important, the fixed-point simulation time becomes prohibitive, and the search space cannot be explored.

An alternative to simulation-based methods is the analytical approach, which determines a mathematical expression of the noise power at the system output according to the statistical parameters of the different noise sources induced by quantization. In this case, the execution time required to evaluate the noise power values is definitely lower. Indeed, the SQNR expression determination is done only once. Then, the SQNR is evaluated quickly at each iteration of the WLO process through a mathematical expression. The method used in this paper to compute the accuracy allows obtaining automatically the quantization noise power expression from the signal flow graph (SFG) of the application. The SFG is obtained from the DFG by inserting the delay operations between data.

An analytical approach, to evaluate the fixed-point accuracy, has been proposed for linear time-invariant systems in [10] and for systems based on smooth operations in [13]. An operation is considered to be smooth if the output is a continuous and differentiable function of its inputs, as it the case for arithmetic operations. In the analytical expression of the output quantization noise power, the gains between the different noise sources and the output are computed from the impulse response of the system between the output and the noise sources. This approach has been implemented in a software tool to automate this process. Our numerical accuracy evaluation tool generates the analytical expression of the output quantization noise from the signal flow graph of the application. This analytical expression is implemented through a C function having the word-length $\mathbf{w}$ of all data as input parameters. This C code can be compiled and dynamically linked to the fixed-point conversion tool for the optimization process.

4.3. Optimization Techniques. In the proposed method, deterministic optimization approach is retained to lead to reasonable optimization times. However, classical greedy algorithms based on steepest-descent (max −1 [14]) or mildest-ascent (min +1 [14]) can lead to weak quality solutions. To improve the solution quality, a tabu search algorithm is used. The proposed method is based on three main steps. First, an initial solution $\mathbf{w}_{\min}$ is determined by computing the minimal word-length associated to each optimization variable $\mathbf{w}(i)$. Then, a mildest-ascent greedy algorithm is used to optimize the word-length. This algorithm starts with the initial solution $\mathbf{w}_{\mathrm{mwc}}$ and leads to the optimized solution $\mathbf{w}_{\min}$. Finally, this optimized solution is refined by using a tabu search algorithm to obtain a better quality solution $\mathbf{w}_{\mathrm{opt}}$.

In the first step, the minimum word-length combination $\mathbf{w}_{\mathrm{mwc}}$ is determined with the algorithm presented in Algorithm 1. For that, all variable word-lengths $\mathbf{w}(i)$ are initially set to their maximal value $w_i^{\max}$. In that case, the accuracy constraint is satisfied. Then, for each variable, the minimum word-length still satisfying the accuracy is determined, all other variable word-lengths staying at their maximum value.

```
for i = 0 ... N_g − 1 do
    w ← (w_0^max ... w_i^max ... w_{N_g−1}^max)
    while λ(w) < λ_min do
        w(i) ← w(i) − 1
    end while
    w_mwc(i) = w(i) + 1
end for
```

Algorithm 1: Determination of the minimal word-length combination.

The mildest-ascent greedy algorithm presented in Algorithm 2 is used to optimize the word-length. Each variable $\mathbf{w}(i)$ is set to its minimal value $\mathbf{w}_{\mathrm{mwc}}(i)$. With this combination, the accuracy constraint will surely not be satisfied anymore. But the advantage of this starting point is that word-lengths only have to be increased to get the optimized solution. At each step of the algorithm, the word-length of one operator is modified to converge to the optimized solution obtained when the accuracy constraint is fulfilled. A criterion has to be defined to select the best direction, that is, the operator for which the word-length has to be modified. The criterion is based on the computation of the discrete gradient of the cost and the accuracy. Let $\nabla_{k/\lambda}$ denote the gradient of the accuracy function

$$\nabla_{k/\lambda} = f_{\mathrm{dir}}(\mathbf{w}_\Delta, \mathbf{w}) = \frac{\lambda(\mathbf{w}_\Delta) - \lambda(\mathbf{w})}{\mathbf{w}_\Delta(k) - \mathbf{w}(k)}, \tag{5}$$

with $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_k, \ldots, \mathbf{w}_{N_g-1})$ and $\mathbf{w}_\Delta = (\mathbf{w}_1, \ldots, \mathbf{w}_k + 1, \ldots, \mathbf{w}_{N_g-1})$.

This gradient on the accuracy is used as a criterion for finding the best direction in the min +1 *bit* algorithm [14]. Amongst deterministic algorithm, min +1 *bit* does not always give a good result. It takes sometimes the wrong direction and returns poor quality results. To improve this criterion, the cost and the accuracy are taken into account as follows:

$$\nabla_k = \frac{\nabla_{k/\lambda}}{\nabla_{k/C}} = \frac{\lambda(\mathbf{w}_\Delta) - \lambda(\mathbf{w})}{C(\mathbf{w}_\Delta) - C(\mathbf{w})}. \tag{6}$$

This criterion selects the direction, which minimizes the cost increase and maximizes the accuracy increase.

Currently, all greedy algorithms used in WLO are monodirection, either steepest-descent (max −1) or mildest-ascent (min +1). To improve the solution obtained with these monodirection algorithms, the proposed algorithm is based on tabu search [15] and allows the movement in both directions.

The set T is the tabu list and contains *tabu variables*. When a variable $\mathbf{w}(k)$ is added in the tabu list, its value will not be modified afterwards and thus this variable is no longer considered in the optimization process. The term d represents the direction, ascending direction is used with $d > 0$, and descending direction is used with $d < 0$. The vector $\mathbf{w}_{\mathrm{opt}}$ corresponds to the best combination of word-lengths

```
w ← w_mwc
while λ(w) < λ_min do
   for i = 0 ... N_g − 1 do
      w_Δ ← w
      w_Δ(i) ← w(i) + 1
      ∇_i ← ((λ(w_Δ) − λ(w)))/((C(w_Δ) − C(w)))
   end for
   k ← argmax_i(∇_i)
   w(k) ← w(k) + 1
end while
w_min = w
```

Algorithm 2: Mildest-ascent greedy algorithm for WL Optimization.

which have been obtained and C_{opt} is the cost associated with $\mathbf{w}_{\text{opt}}$.

The algorithm starts with the solution $\mathbf{w}_{\min}$ obtained with the mildest-ascent greedy algorithm presented in Algorithm 2. This algorithm iterates until all the variable, $\mathbf{w}(k)$ are not in the tabu list (lines 22–23).

For each variable $\mathbf{w}(k)$, the possibility of a movement is analyzed in lines 8–15. If a variable $\mathbf{w}(k)$ reaches its maximal value $\mathbf{w}(k)^{\max}$ in the ascending direction, or its minimal value $b_k^{\min}$ in the descending direction, this variable is added to the tabu list. In the other cases, a movement is possible and the metric for finding the best direction is computed in the lines 16–21. During this metric computation, the cost and the accuracy are compared, respectively, to the best cost C_{opt} and the accuracy constraint $\lambda_{\min}$, and the best solution is updated if necessary.

After the computation of the metric ∇_k for each variable, the best possible direction is selected. For the ascending direction, the solution leading to the highest value of ∇_k is selected (lines 26–28). It corresponds to the solution leading to the best trade-off between the increase of accuracy and the increase of cost. For the descending direction, the solution leading to the lowest value of ∇_k is selected (lines 33–35). The aim is to reduce the cost without reducing too much the accuracy.

As soon as the accuracy constraint is crossed, the direction is inverted (lines 29–31 and 36–38). In this case, the operator is added to the tabu list if the direction is ascending (lines 29–31). This algorithm iterates until all the variables $\mathbf{w}(k)$ are not in the tabu list.

5. Experiments

5.1. Word-Length Optimization Technique. First the quality and the efficiency of the WLO technique based on the tabu search algorithm (Algorithms 1, 2 and Algorithm 3) is evaluated on different benchmarks. The tested applications are a eight-order Infinite Impulse Response filter (IIR) implemented through four second order cells as presented in Figure 6, a Fast Fourier Transform (FFT) on 128 points using a radix-2 and decimation in frequency (DIF) structure and a Normalized Least Mean Square adaptive filter (NLMS) with 128 taps using an adaptation step of 0.5. The implementation cost C_{tabu} and the optimization time T_{tabu} are measured for the proposed technique and compared with the results C_{greedy} and T_{greedy} obtained with only the greedy algorithm corresponding to Algorithms 1 and 2. The number of variables inside the optimization problem is adjusted by grouping together operations. Let I_{tabu} denote the improvement of the solution quality due to the tabu search algorithm such as

$$I_{\text{tabu}} = \frac{C_{\text{greedy}} - C_{\text{tabu}}}{C_{\text{greedy}}}. \tag{7}$$

Let OC_{tabu} denote the over-cost in terms of optimization time due to the tabu search algorithm

$$OC_{\text{tabu}} = \frac{T_{\text{tabu}} - T_{\text{greedy}}}{T_{\text{greedy}}}. \tag{8}$$

For the different experiments, the input signal is normalized in the interval $]-1, 1[$ and different values for the SQNR are tested between 40 to 60 dB by step of 1 dB. The results presented in Table 2 show the improvement obtained with the tabu search algorithm. In our experiments, the improvement can reach up to 65%. The optimization time is significantly increased compared to the greedy algorithm, but the execution time is still reasonable and low compared to other combinatorial optimization approaches like stochastic algorithms.

5.2. Illustrative Example for HLS under Accuracy Constraint. To illustrate the proposed method, an infinite impulse response (IIR) filter example is detailed. This filter is an eight-order IIR filter implemented as four cascaded second-order cells. The signal flow graph (SFG) of this IIR filter, presented in Figure 6, contains 20 multiplications and 16 additions. The method presented in Section 3 is used to obtain the data dynamic range and the binary point-position and thus, a correct fixed-point specification. The SQNR analytical expression is determined and the accuracy constraint is set to 60 dB. The Stratix FPGA is used for the experiments with no dedicated resources.

Firstly, the different operation word-lengths is optimized for a spatial implementation. In this case, an operator is used for each operation. The obtained word-lengths $w_{o_i}^{\text{opt}}$ are presented in Figure 7 (number between parentheses). For the first iteration, a group is defined for each operation type and the group word-lengths are optimized. Thus, multiplications are executed on a 17×17-bit multiplier and additions on 20-bit adders. The minimal system clock frequency is set to 200 MHz, so the operator latency is a multiple of 5 ns. The multiplier and adder propagation times are, respectively, equal to 10.3 ns and 2.5 ns, so the latency of the multiplier and adder is set, respectively, to 3 and 1 clock cycles. The hardware synthesis for this fixed-point specification leads to the scheduling presented in Figure 7. For a 70 ns time constraint, five multipliers and two adders are needed. In the next step, five new groups for multiplications and two new groups for the additions are defined. These groups,

```
(1): w = w_min
(2): C_opt = C(w)
(3): w_opt ← ∅
(4): T ← ∅{tabu operators}
(5): d ← (λ(w) > λ_min) ? − 1 : 1{direction selection}
(6): while |T| < N do
(7):    for all 1 ≤ k ∉ T ≤ N do {computation of ∇k}
(8):       w_Δ ← w
(9):       if d > 0 ∧ w(k) < w^max(k) then
(10):         w_Δ(k) ← w(k) + 1
(11):      else if d < 0 ∧ w(k) > w^min(k) then
(12):         w_Δ(k) ← w(k) − 1
(13):      else
(14):         T ← T ∪ {k}
(15):      end if
(16):      if k ∉ T then {Computation of the metric for direction selection}
(17):         ∇_k ← f_dir(w_Δ, w)
(18):         if λ(w_Δ) > λ_min then
(19):            update of C_opt and w_opt if necessary
(20):         end if
(21):      end if
(22):   end for
(23):   if |T| = N then {all the variable are in the tabu list}
(24):      stop
(25):   end if
(26):   if d > 0 then {selection of the best solution}
(27):      j ← argmax ∇_k
(28):      w(j) ← w(j) + 1
(29):      if λ(w) > λ_min then
(30):         d ← −1{change of direction if λ_min is now fulfilled}
(31):         T ← T ∪ {j}
(32):      end if
(33):   else
(34):      j ← argmin ∇_k
(35):      w(j) ← w(j) − 1
(36):      if λ(w) < λ_min then
(37):         d ← 1{change of direction if λ_min is no longer fulfilled}
(38):      end if
(39):   end if
(40): end while
(41): return w_opt
```

Algorithm 3: Tabu search for solution improvement in WL optimization.

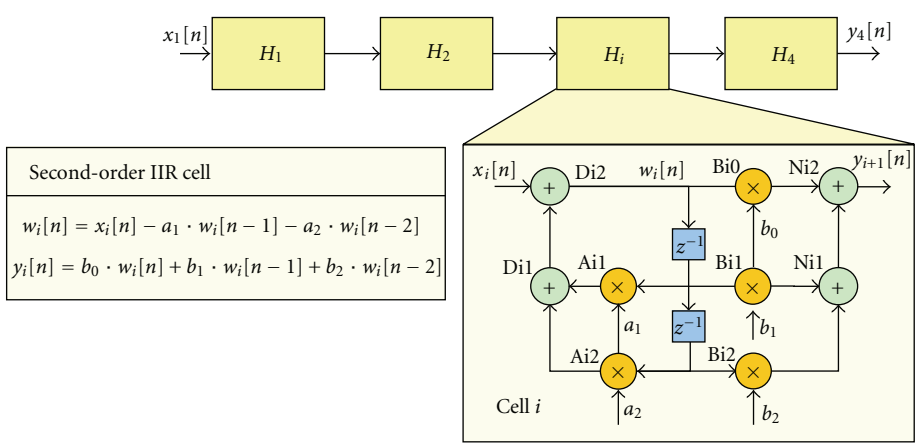

Figure 6: Signal flow graph of the 8th-order IIR filter.

Table 2: Comparison of the tabu search and greedy algorithms in terms of quality improvement I_{tabu} and execution time over-cost (OC_{tabu})

Benchmark	N_g	I_{tabu}	T_{greedy} (s)	T_{tabu} (s)	OC_{tabu}
	14	2.9%	4.5	14.6	219%
IIR	18	6.5%	35.1	83.2	137%
	36	6.6%	78.3	177.1	126%
	8	65.7%	26.1	62.8	141%
FFT	12	62.4%	57.3	163.4	185%
	20	0.6%	57.4	128.8	124%
	13	12.5%	16.8	37.1	120%
NLMS	25	16.3%	76.5	152.4	99%
	49	9.65%	286.5	579.6	102%

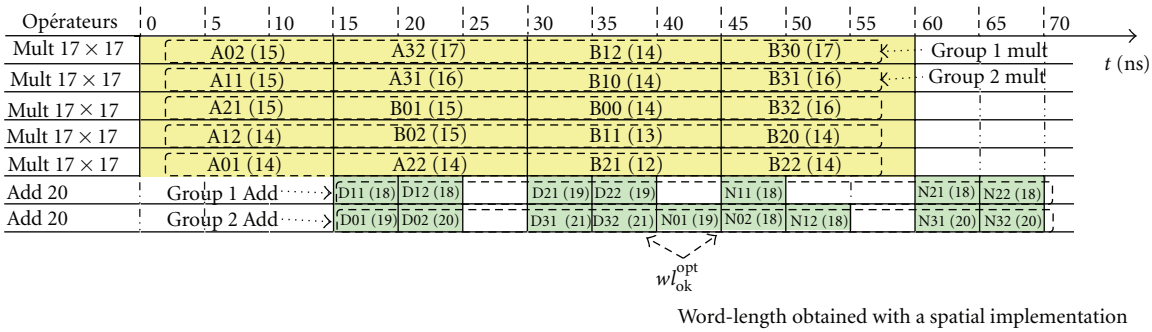

Figure 7: Scheduling of the 8th-order IIR filter obtained for the first iteration.

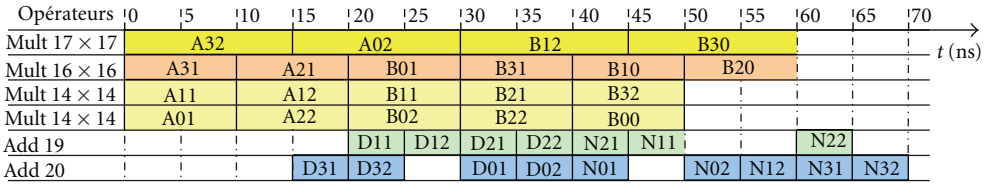

Figure 8: Scheduling of the 8th-order IIR filter obtained for the second iteration.

presented in Figure 7, are built depending on the word-lengths obtained for the spatial implementation and the operation mobility.

A group WLO under accuracy constraint is carried-out for these seven groups. This optimization results in lower word-lengths. The five multiplication group word-lengths are, respectively, 17, 16, 15, 14, and 14 bits. The HLS for this new fixed-point architecture leads to the scheduling presented in Figure 8. Given that, below 16 bits, multipliers have a critical path lower than 10 ns, that is, 2 clock cycles, and so only four multipliers are now needed. Therefore, this architecture uses one less multiplier than the previous one. The word-length reduction combined with the decrease in the number of operators reduces the area by 35%.

A uniform word-length architecture optimization leads to five multipliers and two adders with a precision of 19 bits. Compared to this architecture, the total area saved on operators, with the proposed method, is 47%. A sequential approach carrying-out a word-length optimization in the case of spatial implementation and a high level synthesis leads to the same number of operators as our approach. Nevertheless, the word-length of the operators is higher than those obtain with our approach. Indeed, the operator word-length is imposed by the operation with the greater word-length. Consequently, compared to this sequential approach, the total area saved on operators, with the proposed method, is 6%. These results show the interest of using multiple word-length architecture and efficiency of the proposed method, which couples HLS and WLO.

5.3. Pareto Frontier. The proposed method for multiple word-length HLS generates, for a given timing constraint (latency or throughput) and accuracy constraint, an architecture optimized in terms of implementation cost. By testing different accuracy and timing constraints, the Pareto frontier associated to the application can be obtained. The different trade-off between implementation cost, accuracy, and latency can be analyzed from this curve.

The results obtained for the searcher module of a WCDMA receiver are presented in Figure 9. The data flow graph of the application can be found in [16]. The targeted architecture is an FPGA and only LUTs are considered. The results show an evolution by plateau. For the latency, the plateaus are due to the introduction of one operator or

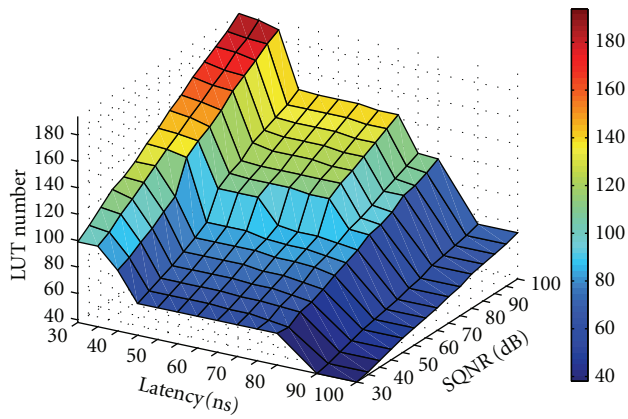

Figure 9: Implementation cost according to the accuracy and timing constraint for WCDMA searcher module.

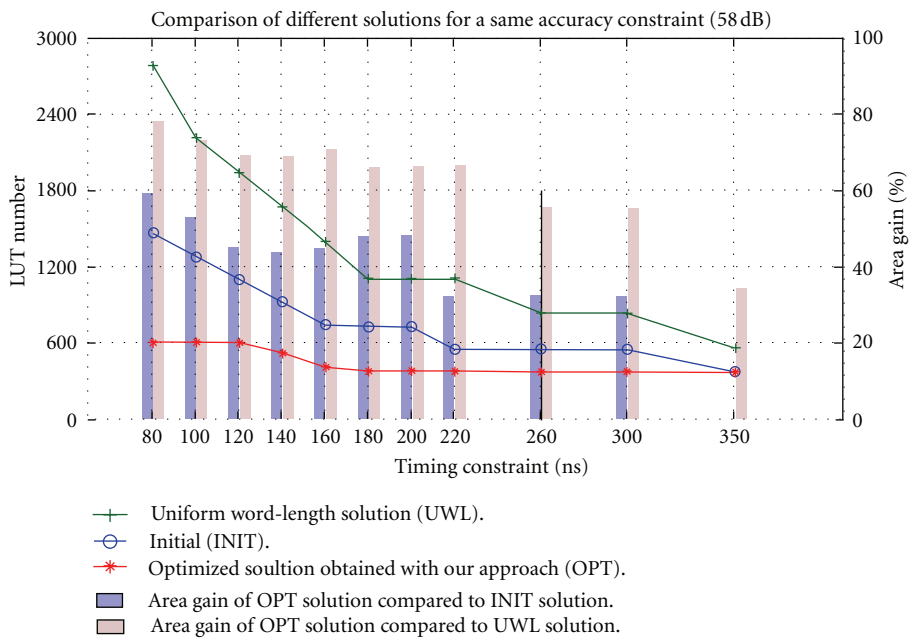

Figure 10: Comparison of the solution obtained by the proposed method (OPT) with a UWL solution and with the initial solution (INIT) for an FFT with an SQNR of 58 dB.

several in parallel, to reduce the application latency. For the accuracy, the evolution is piecewise linear. The smooth evolution is due to the gradual increase of the operator word-length to reach the accuracy constraint. The evolution is linear for this application because the architecture is dominated by addition and subtraction operations, which have a linear implementation cost according to the operand word-length. As for the latency, the abrupt changes in the evolution are due to to the introduction of one operator or several in parallel to reach the constraints. The accuracy increase requires operators with greater word-length and thus leads to higher operator latency. Consequently, when the operator latency increase does not satisfy the global timing constraint, one or more additional operators are required. The location of these abrupt changes in the Pareto frontier is tightly linked to the clock period. The discretization of the operator latency in an integer number of cycles leads to the occurrence of abrupt changes.

5.4. Comparison with Other Solutions. In this section, the solution obtained with the proposed method is first compared with a classical method based on a uniform word-length (UWL) and then with the solution using a single word-length for each type of operation. As in [4, 8], to evaluate the efficiency of the proposed method, the obtained solutions are compared with the UWL solutions. In this last case, a single word-length is used for all data. For a Fast Fourier Transform (FFT), the UWL solution with a 16-bit precision leads to a SQNR of 58 dB. The cost is evaluated with the proposed method (OPT) and with the UWL method for this accuracy constraint of 58 dB and for different timing constraints. The results are presented in Figure 10. For this

Table 3: Area reduction obtained after several iterations compared to the first iteration. Mean and maximal values obtained for different accuracy and timing constraints

Application	$l_{\min}$ (ns)	$\lambda_{\min}$ (dB)	Area Reduction	
			Mean	Max
FIR 32-taps	[100, 200]	[30, 100]	18%	35%
FFT radi-2, DIF, 8 points	[80, 500]	[30, 100]	28%	50%
IIR 8th-*order*	[50, 100]	[30, 100]	22%	47%
Complex correlator	[70, 250]	[10, 100]	10%	20%

application, the proposed method performs better with a gain on the implementation cost between 33% and 78%. When the timing constraint is low, several operators are used for the same kind of operations and the multiple word-lengths approach benefits from the possibility to distribute different word-lengths to each operator. When the timing constraint is high, the number of operators in the architecture is lower and the difference between the OPT and MWL solutions decreases. In the sequential method used in [8, 17], the word-lengths are first optimized and then the architecture is synthesized. The results presented in [8] lead to a gain of up to 52% compared to the UWL solution, and the results presented in [17] leads to a gain of up to 45% compared to the UWL solution. These results show that the combination of the WLO and the HLS in the proposed method gives better results than the sequential method. In [4], the WLO and HLS processes are combined through a simulated annealing optimization and the gain obtained compared to the UWL solution is between 22% and 77%. The proposed method leads to similar gains but with significantly less iterations required for a good solution. Moreover, in our case the HLS process is not modified and existing academic or commercial tools can be directly used.

To analyse the efficiency of the proposed iterative method and the interest of coupling WLO and HLS, the optimized solution obtained after several iterations and the solution obtained at the first iteration are compared. The solution obtained at the first iteration (INIT) corresponds to the case where a single word-length is used for all the operators of the same type. In this case, the operation binding on operators is not taken into account. The architecture area reduction compared to the first iteration is measured and the results obtained for the Stratix FPGA are given in Table 3 for different digital signal processing kernels. The complex correlator computes the correlation between complex signal like in a WCDMA receiver. The experiments are carried out for different accuracy and timing constraints and the maximal and mean values are reported. The proposed method can reduce the architecture area up to 50% in the case of the FFT. In average, the reduction is between 10% and 28%. These results show the efficiency of the iterative method to improve the cost implementation. Gradually, the information collected from the previous iterations allows the convergence to an efficient operation grouping, which improves the HLS.

6. Conclusion

In this paper, a new HLS method under accuracy constraint is proposed. An iterative process is used to link HLS and WLO. This coupling is achieved through an iterative process. To reduce significantly the optimization time compared to the simulation-based methods, the accuracy is evaluated with an analytical method.

The efficiency of proposed method is shown through experiments. Compared to classical implementations based on a uniform word-length, the proposed method reduces significantly the number of resources used to implement the system. These results show the relevance of using multiple word-length architecture. The interest of coupling HLS and WLO is shown on different digital signal processing kernels. This technique reduces the number of operators used in the architecture and also reduces the latency.

References

[1] D. Novo, B. Bougard, A. Lambrechts, L. Van Der Perre, and F. Catthoor, "Scenario-based fixed-point data format refinement to enable energy-scalable software defined radios," in *Proceedings of the IEEE/ACM Conference on Design, Automation and Test in Europe (DATE '08)*, pp. 722–727, Munich, Germany, March 2008.

[2] D. Novo, M. Li, B. Bougard, L. Van Der Perre, and F. Catthoor, "Finite precision processing in wireless applications," in *Proceedings of the IEEE/ACM Conference on Design, Automation and Test in Europe Conference and Exhibition (DATE '09)*, pp. 1230–1233, April 2009.

[3] K. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, pp. 921–930, 2001.

[4] G. Caffarena and C. Carreras, "Architectural synthesis of DSP circuits under simultaneous error and time constraints," in *Proceedings of the 18th IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC '10)*, pp. 322–327, Madrid, Spain, June 2010.

[5] B. Le Gal, C. Andriamisainat, and E. Casseau, "Bit-width aware high-level synthesis for digital signal processing systems," in *Proceedings of the IEEE International SOC Conference (SOCC '06)*, pp. 175–178, September 2006.

[6] P. Coussy, G. Lhairech-Lebreton, and D. Heller, "Multiple word-length high-level synthesis," *Eurasip Journal on Embedded Systems*, vol. 2008, no. 1, Article ID 916867, 11 pages, 2008.

[7] B. Le Gal and E. Casseau, "Latency-sensitive high-level synthesis for multiple word-length DSP design," *Eurasip Journal on Advances in Signal Processing*, vol. 2011, Article ID 927670, 11 pages, 2011.

[8] J. Cong, Y. Fan, G. Han et al., "Bitwidth-aware scheduling and binding in high-level synthesis," in *Proceedings of the ACM/IEEE Asia South Pacific Design Automation Conference (ASP-DAC '05)*, pp. 856–861, Shanghai, China, 2005.

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

[10] D. Menard, R. Rocher, and O. Sentieys, "Analytical fixed-point accuracy evaluation in linear time-invariant systems," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 10, pp. 3197–3208, 2008.

[11] G. Caffarena, G. Constantinides, P. Cheung, C. Carreras, and O. Nieto-Taladriz, "Optimal combined word-length allocation and architectural synthesis of digital signal processing circuits," *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 5, pp. 339–343, 2006.

[12] P. Coussy, C. Chavet, P. Bomel, D. Heller, E. Senn, and E. Martin, *High-Level Synthesis From Algorithm to Digital Circuit, chapter GAUT: A High-Level Synthesis Tool for DSP Applications From C Al-gorithm to RTL Architecture*, Springer, Amsterdam, The Netherlands, 2008.

[13] R. Rocher, D. Menard, P. Scalart, and O. Sentieys, "Analytical accuracy evaluation of Fixed-Point Systems," in *Proceedings of the European Signal Processing Conference (EUSIPCO '07)*, Poznan, Poland, September 2007.

[14] M.-A. Cantin, Y. Savaria, and P. Lavoie., "A comparison of automatic word length optimization procedures," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 2, pp. 612–615, 2002.

[15] F. Glover, "Tabu search—part I.," *INFORMS Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[16] H.-N. Nguyen, D. Menard, R. Rocher, and O. Sentieys, "Energy reduction in wireless system by dynamic adaptation of the fixed-point specification," in *Proceedings of the Workshop on Design and Architectures for Signal and Image Processing (DASIP '08)*, Bruxelles, Belgium, November 2008.

[17] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1432–1442, 2003.