Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2006, Article ID 23197, Pages 1–13
DOI 10.1155/ES/2006/23197

Fixed-Point Configurable Hardware Components

Romuald Rocher, Daniel Menard, Nicolas Herve, and Olivier Sentieys

ENSSAT, Université de Rennes 1, 6 rue de Kerampont, 22305 Lannion; IRISA, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes, France

Received 1 December 2005; Revised 4 April 2006; Accepted 8 May 2006

To reduce the gap between the VLSI technology capability and the designer productivity, design reuse based on IP (intellectual properties) is commonly used. In terms of arithmetic accuracy, the generated architecture can generally only be configured through the input and output word lengths. In this paper, a new kind of method to optimize fixed-point arithmetic IP has been proposed. The architecture cost is minimized under accuracy constraints defined by the user. Our approach allows exploring the fixed-point search space and the algorithm-level search space to select the optimized structure and fixed-point specification. To significantly reduce the optimization and design times, analytical models are used for the fixed-point optimization process.

1. INTRODUCTION

Advances in VLSI technology offer the opportunity to integrate hardware accelerators and heterogenous processors in a single chip (system-on-chip) or to obtain FPGAs with several millions of gate equivalent. Thus, complex signal processing applications can be now implemented in embedded systems. For example, the third generation of mobile communication system requires implementing in a digital platform the wide band code division multiple access (WCDMA) transmitter/receiver, a turbo-decoder, and different codecs for voice (AMR), image (JPEG), and video (MPEG4). The application time-to-market requires reducing the system development time and thus, high-level design tools are needed. To bridge the gap between the available gate count and the designer productivity, design reuse approaches [1] based on intellectual properties (IP) blocks have to be used. The designer assembles predesigned and verified block to realize the architecture.

To reduce the cost and the power consumption, the fixed-point arithmetic has to be used. Nevertheless, the application fixed-point specification has to be determined. This specification defines the integer and fractional word length for each data. The data dynamic range has to be estimated for computing the data binary-point position corresponding to the data integer word length. The fractional part word length depends on the operators word length. For efficient hardware implementation, the chip size and the power consumption have to be minimized. Thus, the goal of this hardware implementation is to minimize the operator word length as long as the desired accuracy constraint is respected.

From an arithmetic point of view, the available IP blocks are limited. In general, the IP user can only configure the input and output word length and sometimes the word length of some specific operators. Thus, the fixed-point conversion has to be done manually by the IP user. This manual fixed-point conversion is a tedious, time-consuming, and error-prone task. Moreover, the fixed-point design search space cannot be explored easily with this approach.

Algorithm level optimization is an interesting and promising opportunity in terms of computation quality. For a specific application, like a linear time-invariant filter, different structures can be tested. These structures lead to different computation accuracy. As shown in the experiment presented in Section 5, for a same architecture the signal-to-quantization-noise ratio (SQNR) can vary from 30 dB to 62 dB for different structures. Thus, this search space must be explored and the adequate structure must be chosen to reduce the chip size and the power consumption. This algorithm level search space cannot be explored easily with available IPs without a huge exploration time. Indeed, the computation accuracy evaluation is based on fixed-point simulations.

In this paper, a new kind of IP optimized in terms of fixed-point arithmetic is presented. The fixed-point conversion is automatically achieved through the determination of the integer and fractional part word lengths. These IPs are configurable according to accuracy constraints influencing

the algorithm quality. The IP user specifies the accuracy constraint and the operator word lengths are automatically optimized. The optimal operator word lengths which minimize the architecture cost and respect the accuracy constraint are then searched. The accuracy constraint can be determined from the application performances through the technique presented in [2].

The computation accuracy is evaluated with analytical approaches to reduce dramatically the optimization time compared to simulation-based approach. Moreover, our analytical approach allows exploring the algorithm level search space in reasonable time.

In this paper, our method is explained through the least mean square (LMS), delayed-LMS (DLMS) applications, and infinite impulse response (IIR) filter. The paper is organized as follows. After a review of the available IP generators, our approach is presented in Section 3. The fixed-point optimization process is detailed in Section 4. Finally, the interest of our approach is underlined with several experiments in Section 5. In each section, the LMS/DLMS application case is developed and the experiments are detailed with IIR applications also.

2. RELATED WORKS

To provide various levels of flexibility, IP cores can be classified into three categories corresponding to hard, soft, or firm cores [1]. Hard IP cores correspond to blocks defined at the layout level and mapped to a specific technology. They are often delivered in masked-level designed blocks (e.g., GDSII format). These cores are optimized for power, size, or performance and are much more predictable. But, they depend on the technology and lead to minimum flexibility. Soft IP cores are delivered in the form of synthesizable register transfer (RT) or behavioral levels hardware description languages (e.g., VHDL, Verilog, or SystemC) code and correspond to the IP functional descriptions. These cores offer maximum flexibility and reconfigurability to match the IP user requirements. Firm IP cores are a tradeoff between the soft and hard IP cores. They combine the high performances of hard cores and the flexibility of soft cores but are restricted in terms of genericity.

To obtain a sufficient level of flexibility, only soft cores are considered in this paper. For soft cores, FPGA vendors often provide a library of classical DSP functions. For most of these blocks, different parameters can be set to customize the block to the specific application [3]. Especially, the data word length can be configured. The user sets these different IP parameters, and the complete RTL code is generated for this configuration. Nevertheless, the link between the application performances and the data word length is not immediate. To help the user to set the IP parameters, some IP providers supply a configuration wizard (Xilinx generator, Altera MegaFunction). The different data word lengths for the IP can be restricted to specific values and all the word lengths cannot be tested. In these approaches, the determination of the binary-point position is not automated and must be done manually by the IP user. This task is tedious, time consuming, and error prone.

The different tools provided by AccelChip integrate an IP generator core (AccelWare) [4] and assist the user to achieve the floating-point to fixed-point conversion [5, 6]. The effect of finite word length arithmetic can be evaluated with Matlab fixed-point simulations. The data dynamic range is automatically evaluated by using the interval arithmetic and the binary-point positions are computed from these information. Then, a fixed-point Matlab code is generated to evaluate the application performances. Thus, the user sets manually the data word length with general rules and modifies them to explore the fixed-point design space. This approach helps the user to convert into fixed-point but does not allow exploring the design space by minimizing the architecture cost under accuracy constraint.

This approach has been extended in [7, 8] to minimize the hardware resources by constraining the quantization error into a specified limit. This optimization is based on an iterative process made up of data word length setting and fixed-point simulations with Matlab. First of all, a coarse grain optimization is applied. In this case, all the data have the same word length. When the obtained solution is closed to the objective, a fine grain optimization is achieved to get a better solution. The different data can have their own word length. This fine grain optimization cannot be applied directly because it will take a long time to converge.

This accuracy evaluation approach suffers from a major drawback which is the time required for the simulation [9]. The simulations are made on floating-point machines, and the extra-code used for emulating the fixed-point mechanisms increases the execution time between one and two orders of magnitude compared to a traditional simulation with floating-point data types [10]. For obtaining an accurate estimation of the noise statistic parameters, a great number of samples must be taken for the simulation. This great number of samples, combined with the increase of execution time due to the fixed-point mechanisms emulation, leads to long simulation time.

This approach becomes a severe limitation when these methods are used in the process of data word length optimization where multiple simulations are needed to explore the fixed-point design space. To obtain reasonable optimization times, heuristic search algorithms like the coarse-grain/fine-grain optimization are used to limit this design space.

Moreover, these approaches test a unique structure for an application. This tool does not explore the algorithm level search space to find the adequate structure which minimizes the chip size or the power consumption for a given accuracy constraint.

3. IP GENERATION METHODOLOGY

3.1. IP generation flow

The aim of our IP generator is to provide an RTL-level VHDL code for an IP with a minimal architecture cost. The architecture cost corresponds to the architecture area, the energy consumption, or the power consumption. This IP generator,

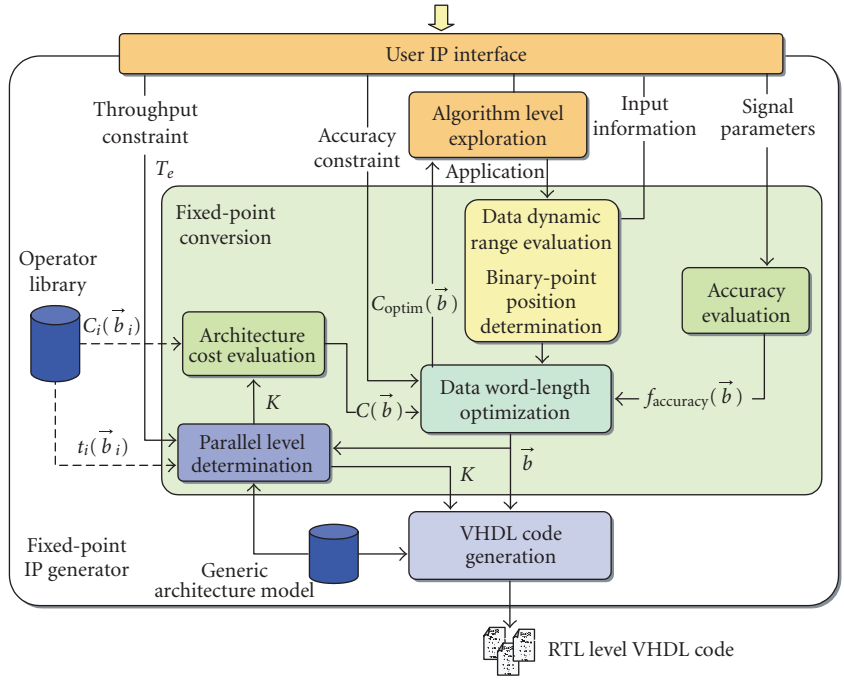

Figure 1: Methodology for the fixed-point IP generation.

presented in Figure 1, is made up of three modules corresponding to the algorithm level exploration, the fixed-point conversion, and the back end which generates the RTL level VHDL code.

The aim of the algorithm level exploration module is to find the structure which leads to minimal architecture cost and fulfils the computation accuracy constraints. This module tests the different structures for a given application, to select the best one in terms of architecture cost. For each structure, the fixed-point conversion process searches the specification which minimizes the architecture cost $C(\vec{b})$ under an accuracy constraint where $\vec{b}$ is the vector containing the data word lengths of all variables. The conversion process returns the minimal cost $C_{\min}(\vec{b})$ for the structure which is selected.

The main part of the IP generator corresponds to the fixed-point conversion process. The aim of this module is to explore the fixed-point search space to find the fixed-point specification which minimizes the architecture cost under accuracy constraints. The first stage corresponds to the data dynamic range determination. Then, the binary-point position is deduced from the dynamic range to ensure that all data values can be coded to prevent overflow. The third stage is the data word length optimization. The architecture cost $C(\vec{b})$ (area, energy consumption) is minimized under an accuracy constraint as expressed in the following expression:

$$\min\left(C(\vec{b})\right) \quad \text{with } \mathrm{SQNR}(\vec{b}) \geq \mathrm{SQNR}_{\min}, \tag{1}$$

where $\vec{b}$ represents all data word length and $\mathrm{SQNR}_{\min}$ the accuracy constraint. The optimization process requires evaluating the architecture cost $C(\vec{b})$ and the computation accuracy $\mathrm{SQNR}(\vec{b})$ defined through the signal-to-quantization-noise ratio (SQNR) metric. This metric corresponds to the ratio between the signal power and the quantization noise power due to finite word length effect. These two processes are detailed in Sections 4.1 and 4.2. To determine the parallelism level K which allows respecting the throughput constraint, the architecture execution time is evaluated as explained in Section 3.3.2. Once the different operator word lengths and the parallelism level are defined, the VHDL code representing the architecture at the RTL level is generated.

3.2. User interface

The user interface allows setting the different IP parameters and constraints. The user defines the different parameters associated with the application. For example, for linear-time-invariant filters, the user specifies the transfer function. For the least-mean-square (LMS) adaptive filter, the filter size or the adaptation step can be specified.

For the fixed-point conversion, the dynamic range evaluation and the computation accuracy require different information on the input signal. The user gives the dynamic range and test vectors for the input signals.

For generating the optimized architecture, the user defines the throughput and the computation accuracy constraints. The throughput constraint defines the output sample frequency and is linked to the application sample frequency. Different computation accuracy constraints can be considered according to the application. For the LMS, the output SQNR is used. For linear-time-invariant filters, three

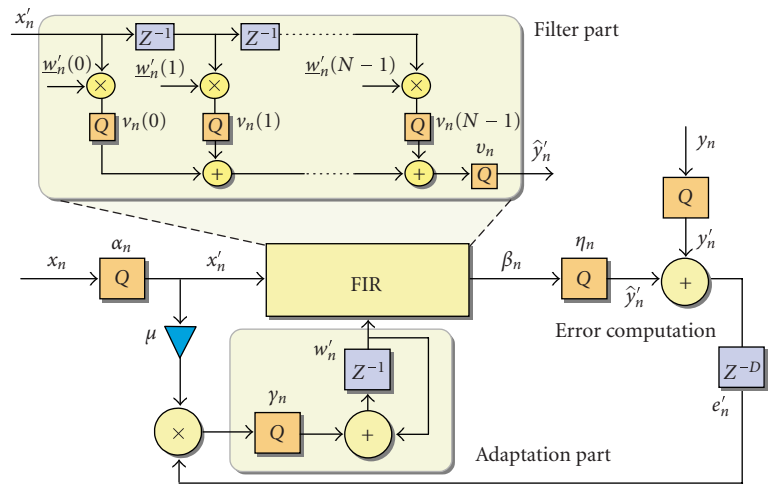

Figure 2: LMS/DLMS algorithm.

constraints are defined. They correspond to the maximal frequency response deviation $|\Delta H_{\max}(\omega)|$ due to finite word length coefficient, the maximal value of the power spectrum for the output quantization noise $|B_{\max}(\omega)|$, and the SQNR minimal value $\mathrm{SQNR}_{\min}$.

3.3. Architecture model

3.3.1. Generic architecture model

Architecture performances depend on algorithm structure. Thus, a generic architecture model is defined for each kind of structure associated with the targeted algorithm. This model can be configured according to the parameters set by the IP user. This architecture model defines the processing and control units, the memory, and the input and output interfaces. The processing unit corresponds to a collection of arithmetic operators, registers, and multiplexors which are interconnected. These operators and the memory are extracted from a library associated with a given technology. The control unit generates the different control signals which manage the processing unit, the memory, and the interface. This control unit is defined with a finite state machine. To explore the search space in reasonable time, analytical models are used for evaluating the architecture cost, the architecture latency, and the parallelism level.

LMS/DLMS architecture

In this part, the architecture of the IP LMS example is detailed. The least-mean-square (LMS) adaptive algorithm, presented in Figure 2, estimates a sequence of scalars y_n from a sequence of N-length input sample vectors $\underline{x}_n$ [11]. The linear estimate of y_n is $\underline{w}_n^t \underline{x}_n$, where $\underline{w}_n$ is an N-length weight vector which converges to the optimal vector $\underline{w}_{\mathrm{opt}}$. The vector $\underline{w}_n$ is updated according to the following equation:

$$\underline{w}_{n+1} = \underline{w}_n + \mu \underline{x}_n e_{n-D} \quad \text{with } e_n = y_n - \underline{w}_n^t \underline{x}_n, \tag{2}$$

where μ is a positive constant representing the adaptation step. The delay D is null for the LMS algorithm and different from zero for the delayed-LMS (DLMS).

The architecture model presented in Figure 3 consists of a filter part and an adaptation part to compute the new coefficient value. To satisfy the throughput constraint, the filter part and the adaptation part can be parallelized. For the filter part, K multiplications are used in parallel and for the adaptation part K multiply-add (MAD) patterns are used in parallel. The different data word lengths $\vec{b}$ in this architecture are b_x for the input filter, b_m for the filter multiplier output, b_h for the filter coefficient, and b_e for the filter output.

To accelerate the computation, the processing is pipelined and the operators work in parallel. Let T_{cycle} be the cycle-time corresponding to the clock period. This cycle-time is equal to the maximum value between multiplier and adder latency. The filter part is divided into several pipeline stages. The first stage corresponds to the multiply operation. To add the different multiplication results, an adder based on a tree structure is used. This tree is made up of $\log_2(K)$ levels. This global addition execution is pipelined. Let L_{ADD} be the number of additions which can be executed in one cycle-time. Thus, the number of pipeline stages for the global addition is given by the following expression:

$$M_{\mathrm{ADD}} = \left\lceil \frac{\log_2(K)}{L_{\mathrm{ADD}}} \right\rceil \quad \text{with } L_{\mathrm{ADD}} = \left\lceil \frac{T_{\mathrm{cycle}}}{t_{\mathrm{ADD\,1}}} \right\rceil, \tag{3}$$

where $t_{\mathrm{ADD\,1}}$ is the 2-input adder latency. The last pipelined stage for the filter part corresponds to the final accumulation. The adaptive part is divided into three pipeline stages. The first one is for the subtraction. The second stage corresponds to the multiplication and the final addition composes the last stage.

3.3.2. Parallelism level determination

To satisfy the throughput constraint specified by the IP user, several operations have to be executed in parallel. The

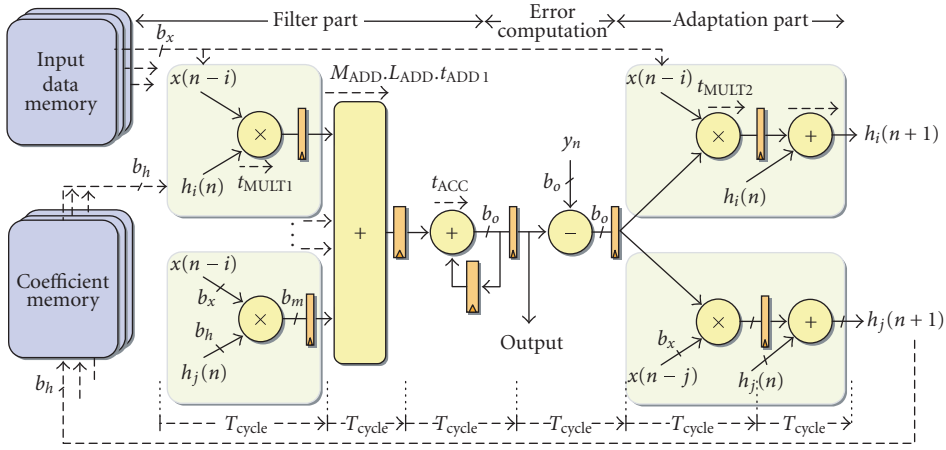

Figure 3: Generic architecture for the LMS/DLMS IP.

parallelism level is determined such that the architecture latency is lower than the throughput constraint. To solve this inequality, the operator latency has to be known and this latency depends on the operator word length. Firstly, the operator word lengths are optimized with no parallelism. The obtained operator word lengths allow determining the operator latency. Secondly, the parallelism level is computed from the throughput constraint, and then the operator word lengths are optimized with the parallelism level real value.

LMS/DLMS architecture

In this part, the architecture of the IP LMS example is detailed. The LMS architecture is divided into two parts corresponding to the filter part and the adaptation part. The execution time of the filter part is obtained with the following expression:

$$T_{FIR} = \frac{N}{K} T_{cycle} + M_{ADD} T_{cycle} + T_{cycle}. \quad (4)$$

The execution time of the adaptation part is given by

$$T_{Adapt} = T_{cycle} + \frac{N}{K}(T_{cycle}) + T_{cycle}. \quad (5)$$

The system throughput constraint depends on the chosen algorithm. For the LMS algorithm, the sampling period T_e must satisfy the following expression:

$$T_{FIR} + T_{Adapt} < T_e. \quad (6)$$

Even if the delayed-LMS algorithm has a slower convergence speed compared to the LMS Algorithm, as the error is delayed, the filter part and the adaptation part can be computed in parallel which gives to the DLMS a potentially higher execution frequency. The constraints become

$$T_{FIR} < T_e, \qquad T_{Adapt} < T_e. \quad (7)$$

The parallelism level is obtained by solving analytically expressions (6) and (7).

3.4. Dynamic range evaluation

Two kinds of method can be used for evaluating the data dynamic range of an application. The dynamic range of a data can be computed from its statistical parameters obtained by a floating-point simulation. This approach estimates accurately the dynamic range with the signal characteristics. Nevertheless, overflow can occur for signals with different statistics. The second method corresponds to analytical approaches which allow computing the dynamic range from input data dynamic range. These types of methods guarantee that no overflow occurs but lead to more conservative results. Indeed, the dynamic range expression is computed in the worst case. The determination of the data dynamic range is obtained by the interval arithmetic theory [12]. The operator output data dynamic range is determined by its input dynamic using propagation rules. For linear time-invariant systems, the data dynamic range can be computed from the $L1$ or Chebychev norms [13] according to the frequency characteristics of the input signal. These norms allow computing the dynamic range of a data in the case of nonrecursive and recursive structures with the help of the computation of the transfer function between the data and each input. For an adaptive filter like the LMS/DLMS, a floating-point simulation is used to evaluate the data dynamic range.

To determine the binary-point position of a data, an arithmetic rule is supplied. The binary-point position m_x of a data x is referenced from the most significant bit as presented in Figure 4. For a data x, the binary-point position is obtained from its dynamic range $\mathcal{D}_x$ with the following relation:

$$m_x = \lceil \log 2(\mathcal{D}_x) \rceil \quad \text{with } \mathcal{D}_x = \max(|x(n)|). \quad (8)$$

A binary-point position is assigned to each operator input and output and a propagation rule is applied for each kind of operators (adder, multiplier, etc.) [14]. Scaling operations are inserted in the graph to align the binary point position in the case of addition or to adapt the binary-point position to the data dynamic range.

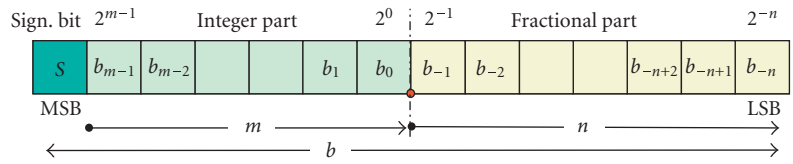

Figure 4: Fixed-point specification.

4. FIXED-POINT OPTIMIZATION

The fixed-point specification is optimized through the architecture cost minimization under a computation accuracy constraint. In this section, the architecture cost and the computation accuracy evaluation are detailed and then, the algorithm used for the minimization process is presented.

4.1. Computation accuracy evaluation

The computation accuracy evaluation based on analytical approach is developed in this part. Quantization noises are defined and modelized, and their propagation through an operator is studied. Then, the expression of the output quantization noise power is detailed for the different kinds of systems.

4.1.1. Noise models

The use of fixed-point arithmetic introduces an unavoidable quantization error when a signal is quantized. A well-known model has been proposed by Widrow [15] for the quantization of a continuous-amplitude signal like in the process of analog-to-digital conversion. The quantization of a signal x is modeled by the sum of this signal and a random variable b_g. This additive noise b_g is a stationary and uniformly distributed white noise that is not correlated with the signal x and the other quantization noises. This model has been extended for modeling the computation noise in a system resulting from the elimination of some bits during a cast operation (fixed-point format conversion), if the number of bits eliminated k is sufficiently high [16, 17].

These noises are propagated in the system through operators. These models define the operator output noise as a function of the operator inputs. An operator with two inputs X and Y and one output Z is under consideration. The inputs X and Y and the output Z are made up, respectively, of a signal x, y, and z and a quantization noise b_x, b_y and b_z. The operator output noise b_z is the weighted sum of the input noises b_x and b_y associated, respectively, with the first and second inputs of the operation. Thus, the function f_γ expressing the output noise b_z from the input noises is defined as follows for each kind of operation γ ($\gamma \in \{+, -, \times, \div\}$) [18]:

$$b_z = f_\gamma(b_x, b_y) = \alpha^{(1)} \cdot b_x + \alpha^{(2)} \cdot b_y. \quad (9)$$

The terms $\alpha^{(1)}$ and $\alpha^{(2)}$ are associated with the noise located, respectively, on the first and second inputs of the operation. They are obtained only from the signal x and y and include no noise term. They are represented on Table 1.

Table 1: Different values of the terms $\alpha^{(1)}$ and $\alpha^{(2)}$ of (9) for different operations $\{+, -, \times, \div\}$.

Operator	Value of $\alpha^{(1)}$	Value of $\alpha^{(2)}$
$Z = X \pm Y$	1	1
$Z = X \times Y$	y	x
$Z = \frac{X}{Y}$	$\frac{1}{y}$	$-\frac{x}{y^2}$

4.1.2. Output quantization noise power

Let us consider, a nonrecursive system made up of N_e inputs x_j and one output y. For multiple-output system, the approach is applied for each output. Let $\hat{y}$ be the fixed-point version of the system output. The use of fixed-point arithmetic gives rise to an output computation error b_y which is defined as the difference between $\hat{y}$ and y. This error is due to two types of noise sources. An input quantization noise is associated with each input $\hat{x_j}$. When a cast operation occurs, some bits are eliminated and a quantization noise is generated. Each noise source is a stationary and uniformly distributed white noise that is uncorrelated with the signals and the other noise sources. Thus, no distinction between these two types of noise sources is done. Let N_q be the number of noise sources. Each quantization noise source b_{q_i} is propagated inside the system and contributes to the output quantization noise b_y through the gain υ_i as presented in Figure 5. The analytical approach goal is to define the power expression of the output noise b_y according to the noise source b_{q_i} parameters and the gains υ_i between the output and the different noise sources.

Linear time-invariant system

For linear time-invariant (LTI) systems, the gain α_i is obtained from the transfer function $H_i(z)$ between the system output and the noise source b_{q_i}. Let $m_{b'_{q_i}}$ and $\sigma^2_{b'_{q_i}}$ be, respectively, the mean and the variance of the noise source b_{q_i}. Thus, the output noise power P_{b_y} corresponding to the second-order moment is obtained with the following expression [13]:

$$P_{b_y} = \sum_{i=0}^{N_q} \sigma^2_{b'_{q_i}} \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_i(e^{j\Omega})|^2 d\Omega + \left(m_{b'_{q_i}} H_i(1)\right)^2. \quad (10)$$

This equation is applied to compute the output noise power of the IIR applications.

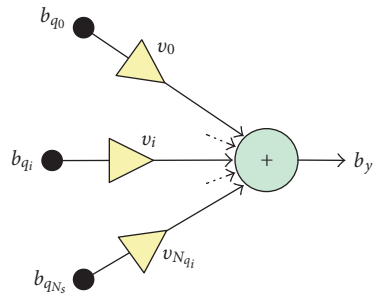

Figure 5: Output quantization noise model in a fixed-point system. The system output noise b_y is a weighted sum of the different noise sources b_{q_i}.

Nonlinear and nonrecursive systems

For the nonrecursive system, each noise b_{q_i} is propagated through K_i operations o_{k_i}, and leads to the b'_{q_i} noise at the system output. This noise is the product of the b_{q_i} input quantization noise source and the different α_k signals associated with each o_{k_i} operation involved in the propagation of the b_{q_i} noise source.

$$b'_{q_i} = b_{q_i} \prod_{k=1}^{K_i} \alpha_k = b_{q_i} v_i \quad \text{with } v_i = \prod_{k=1}^{K_i} \alpha_k. \tag{11}$$

For a system made up of N_q quantization noise sources, the output noise b_y can be expressed as follows:

$$b_y = \sum_{i=0}^{N_s-1} b'_{q_i} = \sum_{i=0}^{N_s-1} b_{q_i} v_i. \tag{12}$$

Given that the b_{q_i} noise source is not correlated with any v_i signal and with the other b_{q_j} noise sources, the output noise power is obtained with the following expression [18]:

$$P_{b_y} = \sum_{i=0}^{N_s} E(b_{q_i}^2) E(v_i^2) + 2 \sum_{i=0}^{N_s} \sum_{\substack{j=0 \\ j>i}}^{N_s} E(b_{q_i}) E(b_{q_j}) E(v_i v_j). \tag{13}$$

The computation of the noise power expression presented in (13) requires the knowledge of the statistical parameters associated with the noise sources b_{q_i} and the signal v_i.

Adaptive systems

For each kind of adaptive filter, an analytical expression of the global noise power can be determined. This expression is established using algorithm characteristics. For gradient-based algorithms, an analytical expression has been developed to compute the output noise power for the LMS/NLMS in [19] and for the affine projection algorithms (APA).

The LMS/DLMS algorithm noise model is presented in the rest of this part. The different noises are presented in Figure 2. With fixed-point arithmetic, the updated coefficient expression (2) becomes

$$\underline{w}'_{n+1} = \underline{w}'_n + \mu e'_n \underline{x}'_n + \underline{\gamma}_n, \tag{14}$$

where $\underline{\gamma}_n$ is the noise associated with the term $\mu e'_n \underline{x}'_n$ and depends on the way the filter is computed. The error in finite precision is given by

$$e'_n = y'_n - \underline{w}'^t_n \underline{x}'_n - \eta_n \tag{15}$$

with η_n the global noise in the inner product $\underline{w}'^t_n \underline{x}'_n$. This global noise is the sum of each multiplication output noise and output accumulation noise:

$$\eta_n = \sum_{i=0}^{N-1} v_n(i) + u_n. \tag{16}$$

Moreover, a new term $\underline{\rho}_n$ is introduced:

$$\underline{\rho}_n = \underline{w}'_n - \underline{w}_n, \tag{17}$$

which is the N-length error vector due to the quantization effects on coefficients. This noise cannot be considered as the noise due to a signal quantization. The mean of each term is represented by m whereas σ^2 represents its variance and can be determined as explained in [17].

The study is made at steady-state, once the filter coefficients have converged. The noise is evaluated at the filter output. The power of the error between filter output in finite precision and in infinite precision is determined. It is composed of three terms:

$$E(b_y)^2 = E(\underline{\alpha}_n^t \underline{w}_n)^2 + E\left(\underline{\rho}_n^t \underline{x}_n\right)^2 + E(\eta_n^2). \tag{18}$$

At the steady state, the vector $\underline{w}_n$ can be approximated by the optimum vector $\underline{w}_{\text{opt}}$. So the term $E(\underline{\alpha}_n^t \underline{w}_n)^2$ is equal to $|\underline{w}_{\text{opt}}|^2 (m_\alpha^2 + \sigma_\alpha^2)$ with $|\underline{w}_{\text{opt}}|^2 = \sum w_{\text{opt}_i}^2$.

The second term $E(\eta_n^2)$ depends on the specific implementation chosen for the filter output computation (filtered data).

The last term is detailed in [19] and is equal to

$$E\left(\underline{\rho}_n^t \underline{x}_n\right)^2 = m_\gamma^2 \frac{\sum_{i=1}^{N} \sum_{k=1}^{N} (R_{ki}^{-1})}{\mu^2} + \frac{N(\sigma_\gamma^2 - m_\gamma^2)}{2\mu}. \tag{19}$$

4.2. Architecture cost evaluation

The IP processing unit is based on a collection of operators extracted from a library. This library contains the arithmetic operators, the registers, the multiplexors, and memory banks for the different possible word lengths. Each library element l_i is automatically generated and characterized in terms of area Ar_i and energy consumption En_i using scripts for the synopsys tools. The IP architecture area (Ar_{IP}) is the sum of the different IP basic element area and the IP memory as explained

Table 2: Different structure complexity for the 8th-order IIR filter.

Kinds of structure	Cell order	Number of cells	Filter complexity			
			Addition	Multiplication	Storage	Coefficients
Direct form I	8	1	16	17	15	17
	4	2	16	18	15	18
	2	4	16	20	15	20
Direct form II	8	1	16	17	12	17
	4	2	16	18	12	18
	2	4	16	20	12	20
Transposed form II	8	1	16	17	12	17
	4	2	16	18	12	18
	2	4	16	20	12	20

in expression (20). Let $\mathrm{IP}_{\mathrm{archi}}$ be the set of all elements setting up the IP architecture. The different element area Ar_i depends on the element word length b_i:

$$\mathrm{Ar}_{\mathrm{IP}}(\vec{b}) = \sum_{l_i \in \mathrm{IP}_{\mathrm{archi}}} \mathrm{Ar}_i\,(b_i). \tag{20}$$

The IP energy consumption ($\mathrm{En}_{\mathrm{IP}}$) is the sum of different operation energy consumption executed to compute the IP algorithm output as explained in expression (21). These operations include the arithmetic operations, the data transfer between the processing unit and the memory (read/write). Let $\mathrm{IP}_{\mathrm{ops}}$, be the set of all operations executed to compute the output. The different En_j operation energy consumption depends on the operation b_j word length

$$\mathrm{En}_{\mathrm{IP}}(\vec{b}) = \sum_{l_j \in \mathrm{IP}_{\mathrm{ops}}} \mathrm{En}_j\,(b_j). \tag{21}$$

The En_j operation energy consumption is evaluated through simulations with synopsys tool. The mean energy is computed from the energy obtained for 10 000 random input data.

4.3. Optimization algorithm

For the optimization algorithm, operations are classified into different groups. A group contains the operations executed on the same operator, and thus these operations will have the same word length corresponding to the operator word length. All group word lengths are initially set to their maximum value. So the accuracy constraint must be satisfied. Then, for each group, the minimum value still verifying the accuracy constraint is determined, whereas all other group word lengths keep their maximum value. Next, all groups are set to their minimum value. The group for which the word length increases gives the highest ratio between accuracy constraint and the cost has its word length incremented until satisfying the accuracy constraint. Finally, all word lengths are optimized under the accuracy constraint.

5. EXPERIMENTS AND RESULTS

Some experiments have been made to illustrate our methodology and to underline our approach efficiency. Two applications have been tested, an 8th-order IIR filter and a 128-tap LMS/DLMS algorithm. The operator library has been generated from 0.18 μm CMOS technology. Each library element is automatically generated and characterized in terms of area and energy consumption using scripts for the synopsys tools.

5.1. IIR filter

5.1.1. IIR IP description

In this part, an infinite impulse response filter (IIR) IP is under consideration. Let N_{IIR} be the filter order. Three types of structure corresponding to direct form I, direct form II, and transposed form II can be used [13]. For high-order filter, cascaded versions have to be tested. The cell order (N_{cell}) can be set from 2 to $N_{\mathrm{IIR}}/2$ if N_{IIR} is even or from 2 to $(N_{\mathrm{IIR}} - 1)/2$ if N_{IIR} is odd. The cell transfer functions are obtained with the factorization of the numerator and denominator polynomials. The complexity of the different IIR filter configurations are presented in Table 2 for an 8th-order IIR filter.

For a cascaded version of the IIR filter, the way that the different cells are organized is important. Thus, different cell permutations must be tested. For the 4th-order cell three different couples of cell transfer functions can be obtained and for each couple, two cell permutations can be tested. For the 2nd-order cell, 24 cell permutations are available. For this 8th IIR filter, the three different types of structure, the different cell orders, the different factorization cases, and the different cell permutations have been tested. It leads to 93 different structures for the same application.

5.1.2. Fixed-point optimization

Coefficient word length optimization

The fixed-point optimization process for the IIR filter is achieved in two steps. First, the coefficient word length b_h is optimized to limit the frequency response deviation $|\Delta H(\omega)|$

due to the finite word length coefficients as in the following equation:

$$\min(b_h) \quad \text{with } |\Delta H(\omega)| \leq |\Delta H_{\max}(\omega)|. \tag{22}$$

The maximal frequency response deviation $|\Delta H_{\max}(\omega)|$ has been chosen such that the frequency response obtained with the fixed-point coefficient remains in the filter template. Moreover, the filter stability is verified with the fixed-point coefficient values. The results obtained for the 8th-order filter obtained with the cascaded and the noncascaded version are presented in Table 3. For high-order cell, the coefficients have greater value, so, more bits are needed to code the integer part. Thus, to obtain the same precision for the frequency response, the coefficient word length must be more important for high-order cell. To simplify, a single coefficient word length is under consideration. Nevertheless, to optimize the implementation, the coefficients associated with the same multiplication operator can have their own word length.

Signal word length optimization

The second step of the fixed-point optimization process corresponds to the optimization of the signal word length. The goal is to minimize the architecture cost under computation accuracy constraints. With this filter IP, two accuracy constraints are taken into account. They correspond to the power spectrum maximal value for the output quantization noise $|B_{\max}(\omega)|$ and the SQNR minimal value $\text{SQNR}_{\min}$

$$\min(C(\vec{b})) \quad \text{with } \begin{matrix} \text{SQNR}(\vec{b}) \geq \text{SQNR}_{\min}, \\ |B(\omega)| \leq |B_{\max}(\omega)|. \end{matrix} \tag{23}$$

The computation accuracy has been evaluated through the SQNR for the 93 structures to analyze the difference between these structures. This accuracy has been evaluated with a classical implementation based on $16 \times 16 \rightarrow 32$- bit multiplications and 32- bit additions. For the noncascaded filter the quantization noise is important and leads to a nonstable filter. The results are presented in Figure 6 for the filter based on 2nd-order cells and 4th-order cells.

The results obtained for the transposed form II are those obtained with the direct form I with an offset. This offset is equal to 7 dB for the filter based on 2nd-order cells, and 9 dB for the filter based on 4th-order cells. These two filter types have the same structure except that the adder results are stored in memory for the transposed form II. This memory storage adds a supplementary noise source. Indeed, in the memory the data word lengths are reduced.

The analysis of the results obtained that for the direct form I and the direct form II, none of these two forms is always better. The results depend on the cell permutations. In the case of filter based on 2nd-order cells, the SQNR varies from 42 dB to 57 dB for the direct form I and from 50 dB to 61 dB for the direct form II. In the case of filter based on 4th-order cells, the SQNR varies from 30 dB to 45 dB for the direct form I and from 26 dB to 49.5 dB for the direct form II. Thus, the choice of filter form cannot be done initially and all the structures and permutations have to be tested.

Table 3: IIR filter coefficient word length.

Cell order	Optimized coefficient word length
8	24
4	15
2	13

The IP architecture area and energy consumption have been evaluated for the different structures and for two accuracy constraints corresponding to 40 dB and 90 dB. The results are presented in Figure 7 for the power consumption and in Figure 8 for the IP architecture area. To underline the IP architecture area variation due to operator word length changes, the throughput constraint is not taken into account in these experiments in the case of IIR filter. Thus, the number of operators for the IP architecture is identical for the different tested structures.

As shown in Figure 6, the filters based on 4th-order cells lead to SQNR with lower values compared to the filters based on 2nd-order cells. Thus, these filters require operator with greater word length to fulfill the accuracy constraint. This phenomenon increases the IP architecture area as shown in Figure 8. Nevertheless, these filters require less operations to compute the filter output. This reduces the power consumption compared to the filters based on 2nd-order cells. Thus, the energy consumption is slightly greater for the filters based on 4th-order as shown in Figure 8.

The energy consumption is more important for the direct form I because this structure requires more memory accesses to compute each filter cell output. For the transposed form II and direct form II, the results are closed. The best solution is obtained for the transposed form II with 2nd-order cells and leads to an energy consumption of 1.6 nJ for the 40 dB accuracy constraint and to 2.7 nJ for the 90 dB accuracy constraint. As shown in Figure 6, this structure gives the lowest SQNR, thus, the operator word length is greater than that for the other forms. Nevertheless, this form consumes less energy because it requires less memory accesses than the direct form II. In the direct form II the memory transfers correspond to the read of the signal to compute the products with the coefficients and the memory write to update the delay taps. In the transposed form II, the memory accesses correspond only to the storage of the adder output.

Compared to the best solution the other structures based on 2nd-order cells leads to a maximal energy over cost of 36% for the 40 dB accuracy constraint and to 53% for the 90 dB accuracy constraint. For the structures based on 4th-order cells the maximal energy over cost is equal to 48% for the 40 dB accuracy constraint and to 71% for the 90 dB accuracy constraint.

The architecture area, is more important for the filters based on 4th-order cells. As explained before, these filters lead to SQNR lower value compared to the filters based on 2nd-order cells. Thus, they require operators with greater word length to fulfill the accuracy constraints. The best solution obtained for the direct form II with 2nd-order cells leads to an architecture area of 0.3 mm^2 for the 40 dB accuracy

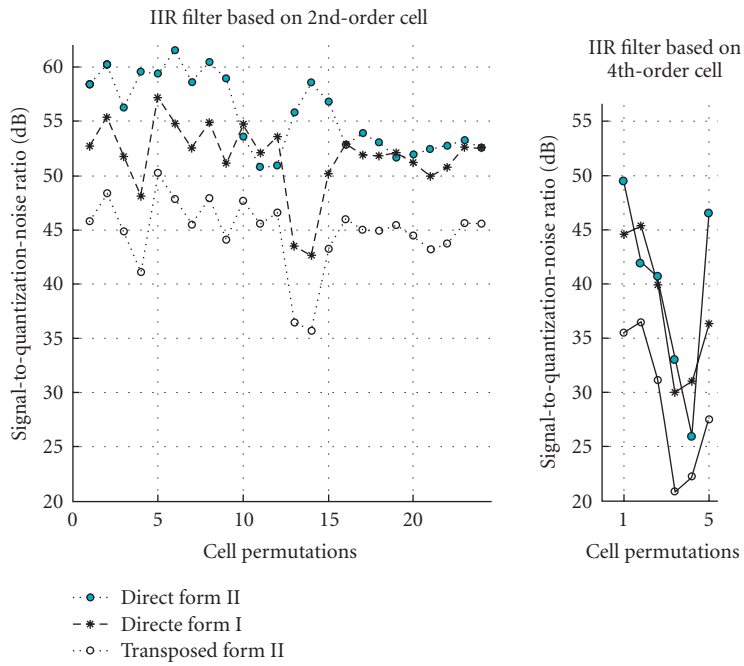

Figure 6: Fixed-point accuracy versus permutations and cell structures.

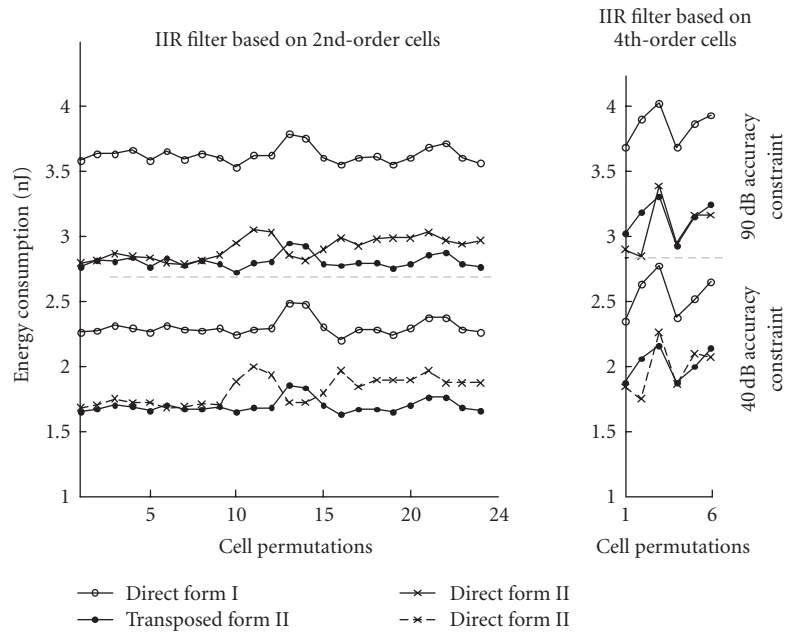

Figure 7: Energy consumption evolution versus cell permutations and cell order.

constraint and to 0.12 mm^2 for the 90 dB accuracy constraint. Compared to this best solution the other structures based on 2nd-order cells lead to a maximal area over cost of 100% for the 40 dB accuracy constraint and to 40% for the 90 dB accuracy constraint. For the structures based on 4th-order cells the maximal energy over cost is equal to 225% for the 40 dB accuracy constraint and to 74% for the 90 dB accuracy constraint.

The best structure depends on the kind of architecture cost. The results are different for the architecture area and for the energy consumption. These results underline the opportunities offered by the algorithm level optimization to

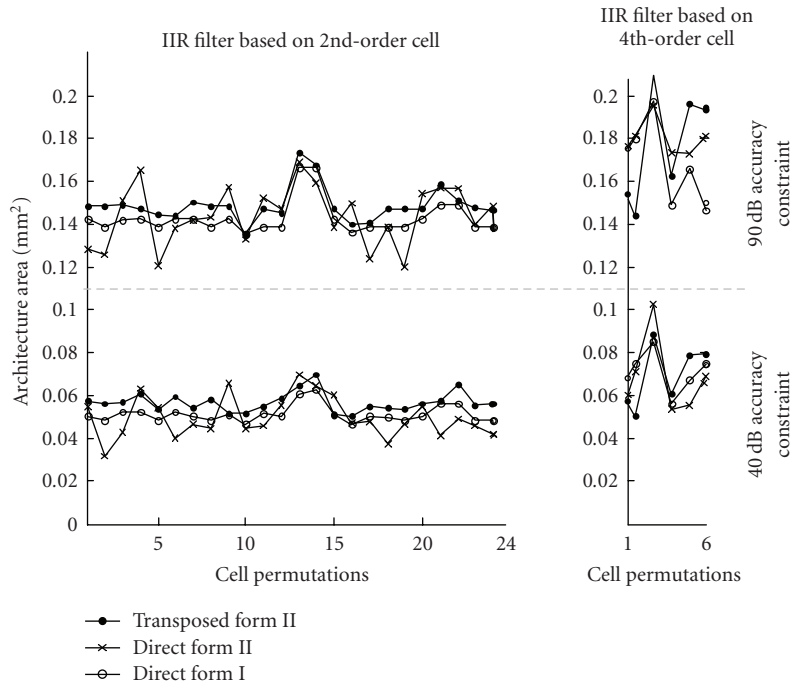

Figure 8: IP architecture area versus permutations cells and cells order.

optimize the architecture cost and the necessity to test the different structures and to select the best one.

For a given structure, the IP architecture area and power consumption evolve linearly according to the SQNR constraint. Between a 40 dB and 90 dB constraints, the energy varies from a factor 1.68 and the area from a factor 4. These results underline the necessity to choose the adapted accuracy constraint in order not to waste energy or area.

5.2. LMS and DLMS algorithms

The LMS and DLMS IP blocks have been used for different experiments to underline the necessity to optimize the operator and memory word lengths under an accuracy constraint. The IP users have to supply the reference and input signal. For the architecture generation, the throughput constraint T_e and the accuracy constraint $\mathrm{SQNR}_{\mathrm{min}}$ must be defined.

The LMS and DLMS IP blocks have been tested for different values of the throughput constraint T_e and the accuracy constraint $\mathrm{SQNR}_{\mathrm{min}}$. For each T_e and $\mathrm{SQNR}_{\mathrm{min}}$ value, the operator and memory word lengths are optimized under the accuracy constraint. Then, the architecture is generated. The architecture area, the parallelism level, and the energy consumption are calculated by simulation and the results are presented, respectively, in Figures 9(a), 9(b), and 9(c) for a timing constraint between 60 ns and 170 ns and for an accuracy constraint between 30 dB and 90 dB.

The evolution of the energy consumption according to the accuracy constraint is presented in Figure 9(c). In our model, the energy consumption is independent of the throughput constraint and the power can be estimated by dividing the energy by the throughput period. The energy consumption varies from 4 nJ to 8.1 nJ for an accuracy constraint going from 30 dB to 90 dB. The energy is multiplied by two between these two accuracy constraints. This energy consumption increase is only due to the growth of the architecture element word length. To fulfill the accuracy constraint, the operator word length has to be increased.

The evolution of the IP architecture area according to the accuracy and throughput constraints are presented in Figures 9(a) and 9(b). For the minimal accuracy and throughput constraints, the architecture area is equal to 0.3 mm^2 with a parallelism level of $K = 4$. The architecture area climbs to 9 mm^2 with a parallelism level of $K = 20$ for the maximal accuracy and throughput constraints. The architecture area increases when the timing constraint decreases. Indeed, to respect this constraint, the parallelism level K must be more important. More operators are needed and thus the processing unit area is increased. The architecture area increases with the accuracy constraint. High values of accuracy constraint require using operators and data with a greater word length. Thus, it increases the energy consumption and the area of the processing and memory units, and moreover, the operator latency. Thus to respect the timing constraint, the parallelism level K must be more important and the processing unit area is increased.

Our results have been compared to a classical solution based on $16 \times 16 \rightarrow 32$-bit multiplications and 32-bit additions. This solution leads to an SQNR of 52 dB. The cost has been evaluated for the classical approach and our optimized approach for an accuracy constraint of 52 dB and with different timing constraints. The results are presented

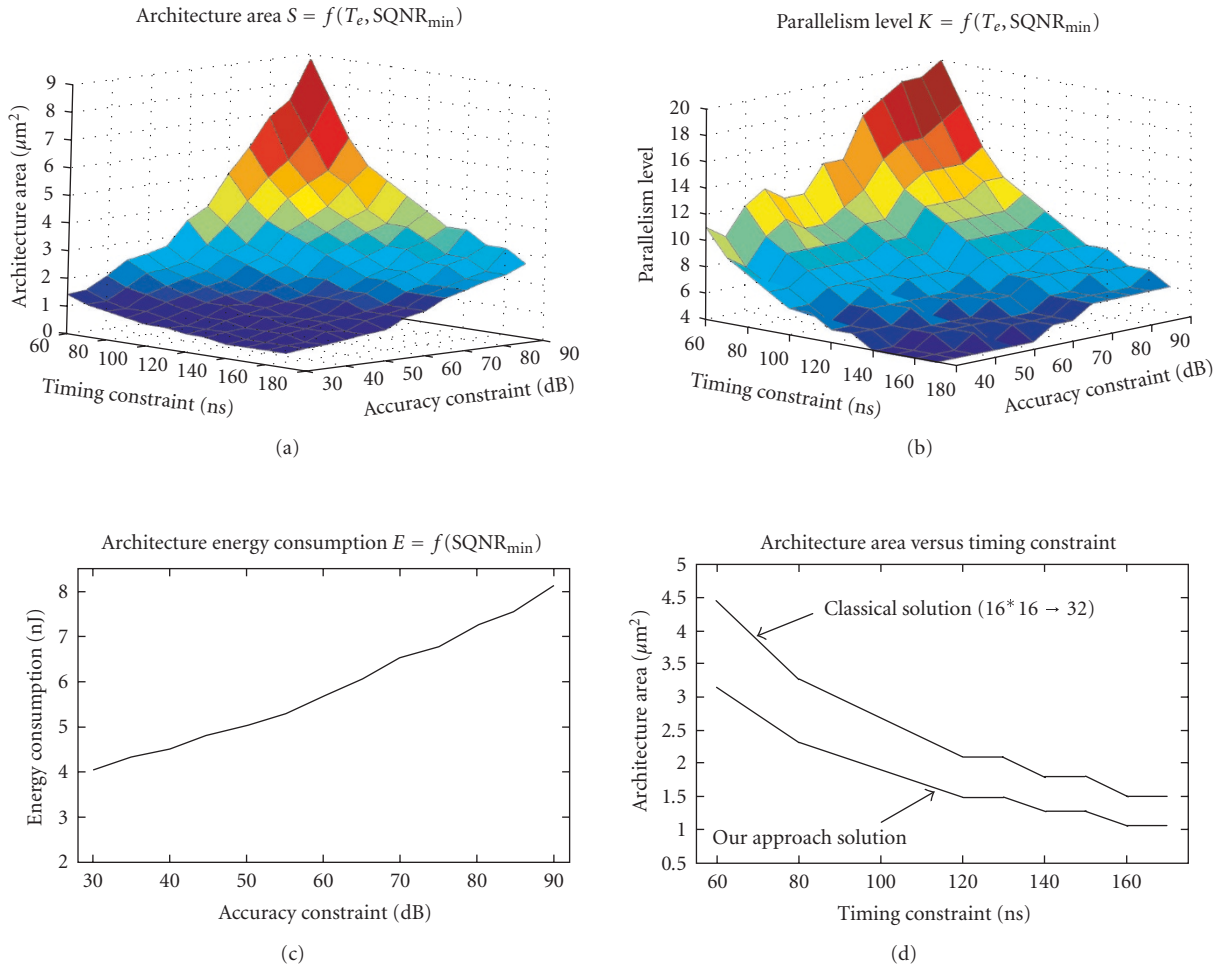

Figure 9: Experiment results: architecture area, energy consumption, and parallelism level for different values of accuracy and timing constraints.

in Figure 9(d). For the same computation accuracy our approach reduces the architecture area by 30% and the power consumption by 23%.

6. CONCLUSION

In this paper, a new kind of fixed-point arithmetic IP has been proposed. The architecture cost is minimized under one or several accuracy constraints. This IP is based on a library of operators and a generic architecture. The parallelism level is adapted to the timing and the computation accuracy constraints. The architecture cost and, especially, the computation accuracy are evaluated analytically to reduce dramatically the evaluation time. This technique allows exploring the fixed-point search space and thus to find the fixed-point specification which optimizes the implementation. Moreover, this analytical approach offers the opportunity to explore the algorithm level search space to find the optimal structure. The results presented for the 8th-order filter underline the interest of algorithm level optimization. The best structure can reduce significantly the IP area and the energy consumption compared to some inefficient structures. For a 128-tap LMS filter, compared to a classical approach, and for the same computation accuracy, the architecture area and the energy consumption are reduced, respectively, by 30% and 23%. With our approach, the user can optimize the tradeoff between the architecture cost, the accuracy, and the execution time.

REFERENCES

[1] M. Keating and P. Bricaud, *Reuse Methodology Manual*, Kluwer Academic, Norwell, Mass, USA, 3rd edition, 2000.

[2] D. Menard, M. Guitton, S. Pillement, and O. Sentieys, "Design and implementation of WCDMA platforms: challenges and trade-offs," in *Proceedings of the International Signal Processing Conference (ISPC '03)*, pp. 1–6, Dallas, Tex, USA, April 2003.

[3] R. Seepold and A. Kunzmann, *Reuse Techniques for VLSI Design*, Kluwer Academic, Boston, Mass, USA, 1999.

[4] AccelChip., "Creating ip for system generator for dsp using matlab," Tech. Rep. 95035, AccelChip, Milpitas, Calif, USA, 2004.

[5] P. Banerjee, D. Bagchi, M. Haldar, A. Nayak, V. Kim, and R. Uribe, "Automatic conversion of floating point MATLAB programs into fixed point FPGA based hardware design," in *Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '03)*, pp. 263–264, Napa Valley, Calif, USA, April 2003.

[6] R. Uribe and T. Cesear, "A methodology for exploring finite-precision effects when solving linear systems of equations with least-squares techniques in fixed-point hardware," in *Proceedings of the 9th Annual Workshop on High Performance Embedded Computing (HPEC '05)*, Lincoln, Neb, USA, September 2005.

[7] S. Roy and P. Banerjee, "An algorithm for converting floating-point computations to fixed-point in MATLAB based FPGA design," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 484–487, San Diego, Calif, USA, June 2004.

[8] S. Roy and P. Banerjee, "An algorithm for trading off quantization error with hardware resources for MATLAB-based FPGA design," *IEEE Transactions on Computers*, vol. 54, no. 7, pp. 886–896, 2005.

[9] L. De Coster, M. Adé, R. Lauwereins, and J. A. Peperstraete, "Code generation for compiled bit-true simulation of DSP applications," in *Proceedings of the 11th IEEE International Symposium on System Synthesis (ISSS '98)*, pp. 9–14, Hsinchu, Taiwan, December 1998.

[10] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: a fixed-point design and simulation environment," in *Proceedings of the IEEE/ACM conference on Design, Automation and Test in Europe (DATE '98)*, pp. 429–435, Paris, France, February 1998.

[11] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 1991.

[12] R. B. Kearfott, "Interval computations: introduction, uses, and resources," *Euromath Bulletin*, vol. 2, no. 1, pp. 95–112, 1996.

[13] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall Signal Processing Series, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 1999.

[14] D. Menard, D. Chillet, and O. Sentieys, "Floating-to-fixed-point conversion for digital signal processors," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 96421, 19 pages, 2006, special issue design methods for DSP systems.

[15] B. Widrow, "Statistical analysis of amplitude quantized sampled-data systems," *Transactions of the American Institute of Electrical Engineer: Part II: Applications and Industry*, vol. 79, pp. 555–568, 1960.

[16] C. W. Barnes, B. N. Tran, and S. H. Leung, "On the statistics of fixed-point round off error," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 3, pp. 595–606, 1985.

[17] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Truncation noise in fixed-point SFGs [digital filters]," *IEE Electronics Letters*, vol. 35, no. 23, pp. 2012–2014, 1999.

[18] D. Menard, R. Rocher, P. Scalart, and O. Sentieys, "SQNR determination in non-linear and non-recursive fixed-point systems," in *Proceedings of the 12th European Signal Processing Conference (EUSIPCO '04)*, pp. 1349–1352, Vienna, Austria, September 2004.

[19] R. Rocher, D. Menard, O. Sentieys, and P. Scalart, "Accuracy evaluation of fixed-point LMS algorithm," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '04)*, vol. 5, pp. 237–240, Montreal, Quebec, Canada, May 2004.

Romuald Rocher received the Engineering degree and M.S. degree in electronics and signal processing engineering from ENSSAT, University of Rennes, in 2003. In 2003, he received the Ph.D. degree in signal processing and telecommunications from the University of Rennes. He is a member of the R2D2 (Reconfigurable Retargetable Digital Devices) Research Team at the IRISA Laboratory. His research interests include floating-to-fixed-point conversion and adaptive filters.

Daniel Menard received the Engineering degree and M.S. degree in electronics and signal processing engineering from the University of Nantes, Polytechnic School, in 1996, and the Ph.D. degree in signal processing and telecommunications from the University of Rennes, in 2002. From 1996 to 2000, he was a Research Engineer at the University of Rennes. He is currently an Associate Professor of electrical engineering at the University of Rennes (ENSSAT) and a member of the R2D2 (Reconfigurable Retargetable Digital Devices) Research Team at the IRISA Laboratory. His research interests include implementation of signal processing and mobile communication applications in embedded systems and floating-to-fixed-point conversion.

Nicolas Herve received the Engineering degree and M.S. degree in signal processing engineering and telecommunications from IFSIC, University of Rennes, in 2002. In 2002, he received the Ph.D. degree in signal processing and telecommunications from the University of Rennes. He is a member of the R2D2 (Reconfigurable Retargetable Digital Devices) Research Team at the IRISA Laboratory. His research interests include floating-to-fixed-point conversion, FPGA architecture, and high-level synthesis.

Olivier Sentieys received the Engineering degree and M.S. degree in electronics and signal processing engineering from ENSSAT, University of Rennes, in 1990, and the Ph.D. degree in signal processing and telecommunications from the University of Rennes, in 1993, and the Habilitation à Diriger des Recherches degree in 1999. He is currently a Professor of electrical engineering at the University of Rennes (ENSSAT). He is the Cohead of the R2D2 (Reconfigurable Retargetable Digital Devices) Research Team at the IRISA Laboratory and is a Cofounder of Aphycare Technologies, a company developing smart sensors for biomedical applications. His research interests include VLSI integrated systems for mobile communications, finite arithmetic effects, low-power and reconfigurable architectures, and multiple-valued logic circuits. He is the author or coauthor of more than 70 journal publications or published conference papers and holds 4 patents.