Word-Length Optimization for Differentiable Nonlinear Systems

GEORGE A. CONSTANTINIDES
Imperial College of Science, Technology and Medicine

This article introduces an automatic design procedure for determining the sensitivity of outputs in a digital signal processing design to small errors introduced by rounding or truncation of internal variables. The proposed approach can be applied to both linear and nonlinear designs. By analyzing the resulting sensitivity values, the proposed procedure is able to determine an appropriate distinct word-length for each internal variable in a fixed-point hardware implementation. In addition, the power-optimizing capabilities of word-length optimization are studied. Application of the proposed procedure to adaptive filters and polynomial evaluation circuits realized in a Xilinx Virtex FPGA has resulted in area reductions of up to 80% (mean 66%) combined with power reductions of up to 98% (mean 87%) and speed-up of up to 36% (mean 20%) over common alternative design strategies.

Categories and Subject Descriptors: B.5.2 [**Register-Transfer-Level Implementation**]: Design Aids—*Automatic synthesis optimization*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Algorithms implemented in hardware*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Signal processing, word-length, bitwidth, synthesis

1. INTRODUCTION

This article is concerned with design automation for digital signal processing (DSP) algorithms implemented in custom parallel hardware. The aim of this work is to raise the level at which DSP systems can be specified beyond a bit-true hardware description language to the domain commonly operated in by digital signal processing (DSP) algorithm designers: an infinite-precision behavioural domain, represented, for example, by the Simulink modelling environment [SIMULINK]. Thus, by using the proposed procedure, a designer need not consider implementation details such as finite word-length effects or fixed-point implementation strategies. The proposed procedure will

The author wishes to acknowledge the financial support of the EPSRC (EP/C512596/1), the Royal Society (2004/R1), and the Nuffield Foundation (URB/01106/G).

Author's address: Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, Exhibition Road, London SW7 2BT, UK; email: g.constantinides@ic.ac.uk.

automatically trade-off implementation area, power, and speed against user-specified numerical accuracy, expressed as a signal-to-noise ratio (SNR).

The accuracy observable at the outputs of a DSP system is a function of the word-lengths used to represent all intermediate variables in the algorithm. However accuracy is less sensitive to some variables than to others, as are implementation area, power consumption, and speed. It is therefore possible to view the design process as a constrained optimization: produce the smallest, lowest power, or fastest implementation subject to constraints on the signal quality at the system outputs.

This article proposes an automated technique for discovering the sensitivity of system outputs to word-length quantization on internal variables, and demonstrates its application in our word-length optimization system, known as Right-Size [Constantinides 2003]. The method presented extends our previous work in the field [Constantinides et al. 2003], which was a fully analytic method only applicable to a restricted class of systems. The result is a hybrid simulation/analytic method, applicable to a broad class of nonlinear system. Two particular nonlinear systems are investigated as case studies: least-mean square adaptive filters, and polynomial evaluation circuits.

The contributions of this article are therefore:

—a novel technique for applying semi-analytic error sensitivity analysis to certain nonlinear systems, extending the domain of applications beyond that addressed by our previous work in this area [Constantinides et al. 2003], which was limited to linear systems;

—an exploration of the power-saving capability of word-length optimization;

—an evaluation of the proposed technique for least-mean-square; (LMS) adaptive filters, showing area reductions of up to 80% combined with power reductions of up to 98% and speed-up of up to 36% over common alternative design strategies;

—an evaluation of the proposed design technique for folded polynomial evaluation architectures, showing area reductions of up to 50% over common alternative design strategies.

The rest of this article is organized as follows. Section 2 reviews the related literature, Section 3 introduces the proposed design flow, and Section 4 describes the required algorithm representation. Section 5 introduces the proposed error analysis procedure, while Sections 6 and 7 examine its application to particular classes of algorithm: LMS adaptive filters, and polynomial evaluation, respectively. Conclusions are drawn in Section 8.

2. BACKGROUND

Several sections of this article will refer to the fundamental signal processing concepts of *linearity* and *time-invariance*. In summary, a system exhibiting the former of these properties is one which responds to the weighted sum of two input sequences with an output sequence equal to the corresponding weighted sum of the individual output sequences. A system exhibiting the latter property is one which responds to a time-shifted input sequence with an equally time-shifted output sequence. Further information on these two system properties may be obtained from any introductory signal processing text, such as Mitra [1998].

In order to obtain an efficient fixed-point implementation of a DSP system while satisfying the computational accuracy constraints imposed by the environment, it is necessary to consider an appropriate *scaling* and *word-length* for each signal. This section reviews the previous work that has taken place in the field of optimization of signal word-length and scaling. It should be mentioned that only work relating to fixed-point or integer arithmetic design is reviewed. There is, however, some ongoing work on floating-point optimization [Gaffar et al. 2004].

Some of the first work to consider these problems was Jackson [1970], where the impact of roundoff noise in linear-system outputs was analysed. In Constantinides and Woeginger [2002] it has been demonstrated that word-length optimization, even for linear time-invariant systems, is NP-hard. There are, however, several published approaches to word-length optimization. Those offering an area/signal quality trade-off propose various heuristic approaches [Constantinides et al. 2003; Wadekar and Parker 1998; Kum and Sung 2001] or do not support different fractional precision for different internal variables [Nayak et al. 2001]. An approach that guarantees optimality in area model, with respect to L_2 error scaling constraints, has also been proposed [Constantinides et al. 2003], however this approach is limited to linear time-invariant systems.

Some published approaches to the wordlength optimization problem use an analytic approach to scaling and/or error estimation [Wadekar and Parker 1998; Nayak et al. 2001; Stephenson et al. 2000; Chang and Hauck 2004], some use simulation [Kum and Sung 2001; Cantin et al. 2001], and some use a hybrid of the two [Keding et al. 1998; Cmar et al. 1999]. The advantage of analytic techniques is that they do not require representative simulation stimulus, and can be faster, however they tend to be more pessimistic. In addition, some published approaches use worst-case instantaneous error as a measure of signal quality [Wadekar and Parker 1998; Nayak et al. 2001; Cantin et al. 2001] whereas some use signal-to-noise ratio (SNR) [Constantinides et al. 2003; Kum and Sung 2001].

In previous work [Constantinides et al. 2003], we have proposed the use of analytic techniques for word-length and scaling optimization, achieving up to 45% area reduction combined with 39% speed-up; the main limitation of this work is its applicability only to linear time-invariant systems. The proposals in this paper use semi-analytic error models to extend the previous work to systems containing differentiable nonlinearities, such as general multipliers.

3. DESIGN FLOW

Before discussing the internal details of the Right-Size word-length optimization system, it is first appropriate to consider the use of the system within a familiar design flow, in order to place the tool in context.

DSP algorithm design is often initially performed directly in a graphical programming environment such as Mathworks' Simulink [SIMULINK]. Simulink

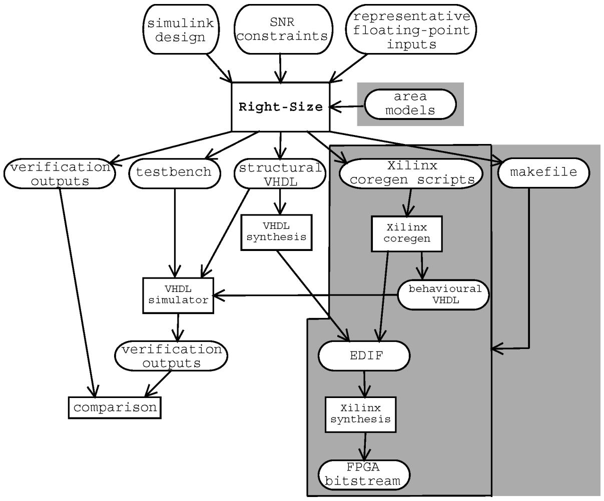


Fig. 1. Design Flow (shaded portions are technology-specific, and are illustrated for Xilinx FPGAs).

is widely used within the DSP community and, for FPGA-based design, has been recently incorporated into the Xilinx System Generator [Hwang et al. 2001] and Altera DSP Builder design flows. Similar tools for application-specific integrated circuit (ASIC) flows include the Signal Processing Worksystem from Cadence.

It is usual that for custom hardware implementations, such descriptions must be annotated by the used with hardware-specific features, such as signal word-length and scaling (binary point location). The advantage of the proposed approach is that the engineer need not specify these features; beyond a standard Simulink algorithm description, only one piece of information is required: a lower-bound on the *output* signal to quantization noise (SNR) acceptable to the user. The proposed design approach thus represents a truly "behavioral" synthesis route, exposing to the DSP engineer only those aspects of design naturally expressed in the DSP application domain.

A diagram of the design flow for Xilinx FPGAs is shown in Figure 1, with the technology-specific portions shaded. The proposed technique is entirely technology independent, although the exact savings possible by using the approach may vary from technology to technology. In order to retarget the synthesis tool to a different technology it is sufficient to create a new module-generation method for each type of computation supported, and characterize the modules with parameterizable area models.

As shown in Figure 1, the inputs to the proposed approach are a specification of the system behavior (e.g. using Simulink), a specification of the acceptable SNR at each output, and a set of representative input signals. From these inputs, the tool automatically generates a synthesizable structural description of the architecture and a bit-true behavioral VHDL testbench, together with a set of expected outputs for the provided set of representative inputs. Also generated is a makefile which can be used to automate the back-end synthesis process.

In general, there is a sensitivity of the resulting synthesised architecture to the choice of representative input data, which manifests itself in two forms. First, the input signals used should be sufficient to exercise the datapath to within a factor of the full dynamic range required, otherwise unwanted overflow errors could occur on other data sets encountered in practice. Second, the quantization error produced by the resulting system could break the user-specified bounds if the system is driven by inputs with wholly different statistical properties, although the bounds are guaranteed for the specific set of data provided.

4. ALGORITHM REPRESENTATION

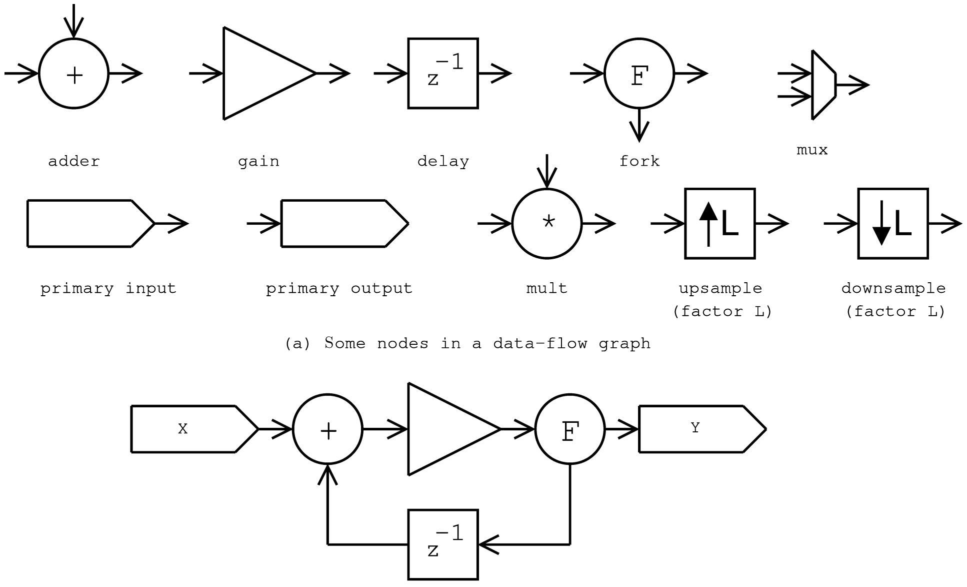
Individual computations in a design can be one of many types. The proposed procedure requires each of these computation types to produce a differentiable function of its inputs, apart from quantizers, which may be incorporated into the quantization noise-model proposed in this article. This is not a major restriction for DSP algorithms, which tend to consist of multiplication, addition, and continuous function evaluation. For control-intensive applications commonly found, for example, in network applications, the proposed technique is inapplicable to the system as a whole. It is always possible, however, to partition a hybrid system into portions of DSP datapath logic, to which the proposed technique can be applied on an individual basis.

The representation used as a starting point for the optimization techniques described in this article is referred to as a data-flow graph (DFG) [Lee and Messerschmitt 1987]. A DFG $G(V, S)$ is the formal representation of an algorithm. V is a set of graph nodes, each representing an atomic computation or input/output port, and $S \subset V \times V$ is a set of directed edges representing the data flow. An element of S is referred to as a *signal*.

Throughout this article, DFGs will be visualized using a graphical representation, as shown in Figure 2. Note that the multiplexer node is illustrated without a select line input. This is because this input is assumed to come from some external controller, decoupled from the datapath. For systems where the select inputs depend on computed values in the datapath, the proposed procedure is inapplicable, as this is a nondifferentiable nonlinear dependence.

5. PROPOSED PROCEDURE

The proposed procedure consists of three steps. First, a perturbation analysis is performed, in order to quantify the sensitivity of each system output to noise at each point within the computational structure. Second, a scaling analysis is performed in order to determine an appropriate binary-point location for each signal in the system. Finally, the results of the two preceding steps are used in a word-length optimization procedure that aims to find the lowest area implementation of the system, subject to the user's specified constraints on signal to noise ratio. The first two of these three steps are described below. The third step is then only briefly reviewed, as it is identical to that published previously and shown to produce near-optimal results for small systems [Constantinides et al. 2003]. All steps have been fully automated within the working synthesis system described in Section 3, and may be applied to any



(a) Some nodes in a data-flow graph

(b) An example data-flow graph

Fig. 2. The graphical representation of a data-flow graph.

DFG consisting of the node types shown in Figure 2 or, *mutatis mutandis*, to any system containing linear and differentiable nonlinear blocks only.

5.1 Perturbation Analysis

In order to make some of the analytical results on error sensitivity for linear, time-invariant systems [Constantinides et al. 2003] applicable to nonlinear systems, the first step is to linearize these systems. The assumption is made that the quantization errors induced by rounding or truncation are sufficiently small not to affect the macroscopic behavior of the system (the so-called *small perturbation hypothesis* [Alippi 2002]). Under such circumstances, each component in the system can be locally linearized, or replaced by its “small-signal equivalent” [Sedra and Smith 1991] in order to determine the output behavior under a given rounding scheme. It should be noted that this is not a strong assumption; from a user’s perspective, adoption of this assumption corresponds to a mis-estimation of the signal to noise ratio (SNR) for extremely low values of SNR, a situation we have not experienced in practice.

Consider one such n-input component: the differentiable function $Y[t] = f(X_1[t], X_2[t], \ldots, X_n[t])$, where t is a time index. If we denote by $x_i[t]$ a small perturbation on variable $X_i[t]$, then a first-order Taylor approximation for the induced perturbation $y[t]$ on $Y[t]$ is given by $y[t] \approx x_1[t]\frac{\partial f}{\partial X_1} + \cdots + x_n[t]\frac{\partial f}{\partial X_n}$. The accuracy of the resulting approximation is dependent on the contribution of the higher-order Taylor terms, but at high values of SNR, the approximation is in practice highly accurate, as the peak value of $x_i[t]$ falls exponentially with the number of bits used to represent $X_i[t]$, at the rate of 6 dB per bit.

Table I. Node Properties

Classification	Node Type	Function	Taylor Coefficients
linear, time-invariant	adder	$Y = f(X_1, X_2) = X_1 + X_2$	$\frac{\partial f}{\partial X_1} = 1$, $\frac{\partial f}{\partial X_2} = 1$
	gain	$Y[t] = f(X_1[t]) = \alpha X_1[t]$	$\frac{\partial f}{\partial X_1} = \alpha$
nonlinear, time-invariant	mult	$Y[t] = f(X_1[t], X_2[t]) = X_1[t]X_2[t]$	$\frac{\partial f}{\partial X_1} = X_2[t]$, $\frac{\partial f}{\partial X_2} = X_1[t]$
linear, time varying	mux	$Y[t] = f_t(X_1[t], X_2[t]) = X_{\mathrm{sel}(t)}[t]$	$\frac{\partial f_t}{\partial X_i} = \begin{cases} 1, \mathrm{sel}(t) = i \\ 0, \text{otherwise} \end{cases}$

Note that this approximation is linear in each x_i, but that the coefficients may vary with time index t since in general $\frac{\partial f}{\partial X_i}$ is a function of $X_1, X_2, \ldots, X_n$. Thus by applying such an approximation, a linear time-varying small-signal model has been produced for a nonlinear time-invariant component. This approach may also be generalized to nonlinear time-varying components in a straight-forward manner by considering the function to have the form $Y[t] = f_t(X_1[t], X_2[t], \ldots, X_n[t])$. The proposed technique is thus also suitable for time-varying systems, as the Taylor coefficients are themselves time-varying.

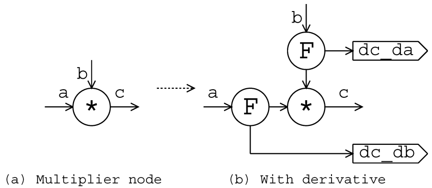
The linearity of the resulting model allows the prediction of the error at system outputs due to any scaling of a small perturbation of signal $s \in S$ analytically, given the simulation-obtained error of a *single* instance of the perturbation at s. Thus, the proposed method can be considered to be a hybrid analytic/simulation error analysis.

Simulation is performed at several stages of the analysis, as detailed below. In each case, our tool Right-Size takes advantage of the static schedulability of the synchronous data-flow model implied by the algorithm representation, leading to a fast simulation compared to an event-driven method [Lee and Messerschmitt 1987]. The entire perturbation analysis execution time scales linearly with the size of the representative data set, and quadratically with the size of the system, as each derived linearized system needs to be simulated for each possible point of noise injection.

5.1.1 *Preliminaries.* While the approach described in this article is a general one, only the node types presented in Figure 2 are discussed in this article. These node types may be classified according to their linearity and time-properties, as shown in Table I for the combinational nodes of Figure 2.

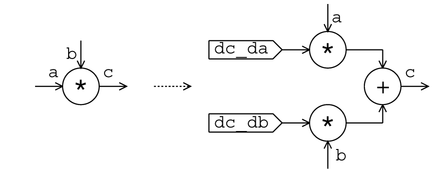
5.1.2 *Derivative Monitors.* In order to construct the small-signal model, the tool must first evaluate the differential coefficients of the Taylor series model for nonlinear components. Like other procedures described in this paper, this process is expressed as a graph transformation.

In general, methods must be introduced to calculate the differential of each nonlinear node type. This is performed by applying a graph transformation to the DFG, introducing the necessary extra nodes and outputs to calculate this



(a) Multiplier node (b) With derivative monitors

Fig. 3. Local graph transformation to insert derivative monitors.



(a) Multiplier node (b) Its first-order Taylor model

Fig. 4. Local graph transformation to produce small-signal model.

differential. As an example, the graph transformation for multipliers, implementing the description in Table I, is illustrated in Figure 3.

After insertion of the monitors, the DFG is simulated to provide the (double precision floating-point) derivatives required by the linearization process, to be described below.

5.1.3 *Linearization.* The construction of the small-signal model may now proceed, again through graph transformation. All linear components (such as adder, constant-coefficient multiplier, fork, delay, primary input, and primary output) remain unchanged as a result of the linearization process. Each nonlinear component is replaced by its truncated Taylor series model. Additional primary inputs are added to the DFG to read the Taylor coefficients from the derivative outputs created by the above large-signal simulation. As an example, the Taylor expansion transformation for the multiplier node is illustrated in Figure 4.

5.1.4 *Noise Injection.* The SNR measure of signal quality is used to specify the user-acceptable limits on signal distortion through truncation or rounding. In [Constantinides et al. 2003], so-called L_2-scaling was used to analytically estimate the noise variance at a system output through scaling of the (analytically derived) noise variance injected at each point of quantization. Such a purely analytic technique can be used only for linear time-invariant systems, however in this paper an extension of the approach for nonlinear systems is suggested.

Since the small-signal model is linear, if an output exhibits variance V when excited by an error of variance σ^2 injected into any given signal, then the output will exhibit variance αV when excited by a signal of variance $\alpha\sigma^2$ injected into the same signal ($0 < \alpha \in \mathbb{R}$). Herein lies the strength of the proposed linearization procedure: if the output response to a noise of known variance

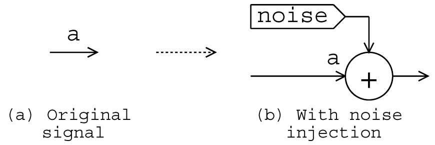


Fig. 5. Local graph transformation to inject perturbations.

can be determined *once only* through simulation, this response can be scaled with analytically derived coefficients in order to estimate the response to any rounding or truncation scheme. One of the great advantages of this approach is the independence of the computational complexity to the absolute level of rounding error; pure simulation-based approaches require large data-sets for high signal-to-noise ratio specifications.

Thus, the next step of the procedure is to transform the graph through the introduction of an additional adder node, and associated signals, and then simulate the graph with a known noise. To simulate truncation of a two's complement signal, the noise is independent and identically distributed with a uniform distribution over the range $[-2\sqrt{3}, 0]$. This range is chosen to have unit variance, thus making the measured output response an unscaled sensitivity measure.

The graph transformation of inserting a noise injection is shown in Figure 5. One of these transformations is applied to a distinct copy of the linearized graph for each signal in the DFG, after which zeros are propagated from the *original* primary-inputs, to finalize the small-signal model. This is a special case of constant propagation [Aho et al. 1986] which leads to significantly faster simulation results for nontrivial DFGs.

The entire process is illustrated for a simple DFG in Figure 6. The original DFG is illustrated in Figure 6(a). The perturbation analysis is performed for the signals marked (*) and (**) in this figure. After inserting derivative monitors for nonlinear components, the transformed DFG is shown in Figure 6(b). The linearized DFG is shown in Figure 6(c), and its two variants for the signals (*) and (**) are illustrated in Figures. 6(d) and (e) respectively. Finally, the corresponding simplified DFGs after zero-propagation are shown in Figures. 6(f) and (g) respectively.

5.2 Scaling Analysis

Each signal in a multiple word-length system has two parameters, its word-length n and its scaling p, as introduced in Constantinides et al. [2003] and illustrated in Figure 7.

Scaling analysis is an important research topic in its own right [Constantinides et al. 2004]. As the main focus of this work is on word-length optimization, the proposed approach uses a relatively simple scaling technique combining the approach of [Kum and Sung 2001] with [Hwang et al. 2001]. The approach taken is described below for completeness, but could be substituted by any other approach, such as recent affine arithmetic techniques [Fang et al. 2003], without modifying the proposed word-length optimization procedure.

Scaling analysis is performed in two phases. First, a simulation is performed using the user-provided input sequence. The peak signal value P_s reached by

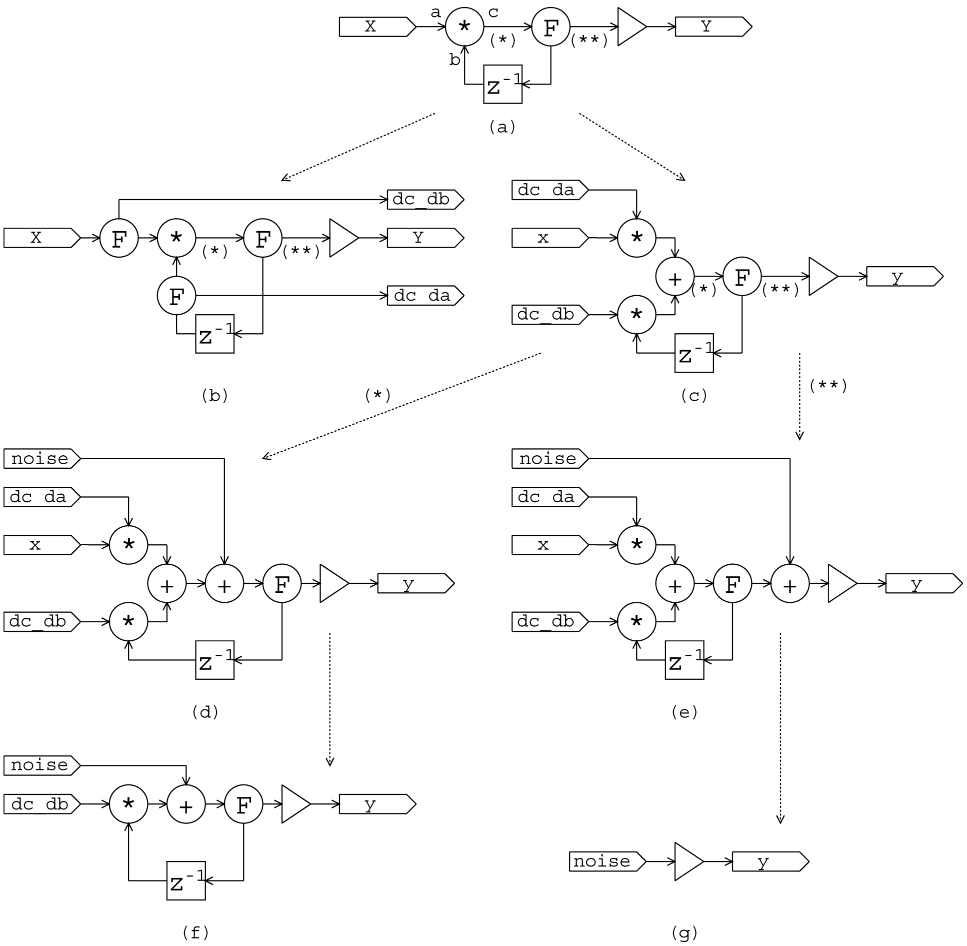


Fig. 6. Example perturbation analysis.

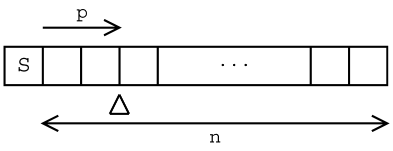


Fig. 7. A multiple word-length signal (“S” represents the sign bit), the triangle represents the binary-point location.

each signal $s \in S$ in the simulation is recorded, and scaled-up by a user-defined ‘safety factor’ k (typically $k = 4$, i.e., two most-significant guard bits). The scaling $p_s = \lfloor \log_2 kP_s \rfloor + 1$ is thus derived [Constantinides et al. 2004]. For some systems, the safety factor may lead to an overly pessimistic scaling, and so the role of the second phase is to compensate for this.

The second phase determines the maximum scaling required at the output of each node, in terms of the scalings at the input to each node. For example, if an adder has inputs with scaling p_a and p_b, then its output cannot have a scaling above $\hat{p}_s = \max(p_a, p_b) + 1$. If this scaling is less than the simulation-determined version, that is, $\hat{p}_s < p_s$, then p_s is set equal to $\hat{p}_s$. Such a 'conditioning' is performed for each node in the DFG, and if there have been any changes, the process is repeated until the scalings converge.

5.3 Word-Length Optimization

The word-length optimization procedure used in Right-Size is identical to that described for linear time-invariant systems in Constantinides et al. [2003].

By using the perturbation sensitivities calculated as described in Section 5.1, estimation of the error at system outputs, given a scaling and word-length for each signal, is a trivial operation, requiring only the calculation of the error variance injected at each signal as described in [Constantinides et al. 2003]. This allows the optimization procedure to obtain a fast on-the-fly estimation of signal quality.

The exact procedure used for area estimation, given a scaling and word-length for each signal, will depend on the target technology. Currently, Right-Size targets Xilinx Virtex FPGAs [Xilinx, Inc. 2002], and instantiates cores from the Xilinx Coregen system to implement the integer arithmetic units at the heart of each multiple word-length component. By a thorough analysis of these cores, involving the synthesis of several thousand parameter values, high-level models of LUT-usage have been constructed, enabling the word-length optimization procedure to obtain a fast on-the-fly estimation of resource usage.

6. CASE STUDY 1: ADAPTIVE FILTERING

Adaptive filtering is a common DSP application, especially in the field of communications where it is widely used, for example, to compensate for multipath distortion in mobile communication systems [Haykin 1996].

In addition to its practical significance, adaptive filtering has some interesting algorithmic features:

- All adaptive filtering algorithms contain feedback, limiting the applicability of several existing word-length optimization techniques [Wadekar and Parker 1998; Nayak et al. 2001; Stephenson et al. 2000; Cmar et al. 1999] and limiting the performance achievable through pipelining.
- Adaptive filters contain general multipliers, rather than the constant coefficient multipliers present in static filters. This means that adaptive filters are nonlinear systems, limiting the applicability of purely analytic techniques such as Constantinides et al. [2003].
- The coefficients of an adaptive filter are updated by accumulating (usually small) correction terms. Such "integration loops" make the outputs of an adaptive filter very sensitive to errors induced around such loops.

The least-mean-square (LMS) adaptive filter [Haykin 1996] is considered in this section, due to its widespread use in practice. For the unfamiliar reader, a brief review of LMS filters is first provided.

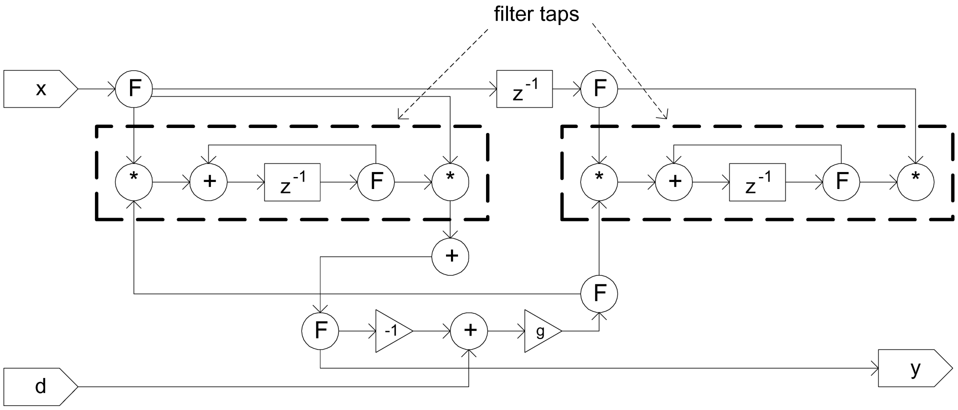


Fig. 8. First order LMS adaptive filter.

6.1 LMS Filters: A Review

Consider an input signal $x[t]$ and a desired filter response $d[t]$. (The desired response could be known *a-priori*, for example from a 'training sequence' used in GSM mobile telephony). Let n denote the *order* of the filter, and $\mathbf{u}[t]$ denote the vector $\mathbf{u}[t] = (x[t]\ x[t-1]\ x[t-2]\ \cdots\ x[t-n])^T$, where ^{T} represents vector transpose. An LMS filter with real input and coefficients has the following algorithm, where $\mathbf{0}$ represents a column vector with each element equal to 0, and g is a user-chosen scalar adaptation coefficient.

```
$\mathbf{w}[0] = \mathbf{0}$
for $t \geq 0$ do
    $y[t] = \mathbf{w}^T[t]\mathbf{u}[t]$
    $e[t] = d[t] - y[t]$
    $\mathbf{w}[t+1] = \mathbf{w}[t] + g\mathbf{u}[t]e[t]$
end do
```

A DFG for a first-order LMS filter is shown in Figure 8. The DFG for an nth order filter is easily derived through a replication of the taps and the use of an adder-tree to sum the partial results.

6.2 Results

In order to demonstrate the area, power, and delay advantages of the proposed method, 90 filters of between 1st and 10th order have been constructed and passed to the Right-Size tool for synthesis. In each case the "desired" input $d[t]$ to the adaptive filter is a well-known 100,000 sample voice clip from FREETEL [1993]. The filter input $x[t]$ is a version of the same signal, corrupted by three different 12th order autoregressive filters, operating on three disjoint and equally sized portions of the input signal. Each distortion filter has constant coefficients randomly chosen such that the filter poles occur in complex conjugate pairs and have independent, identically distributed uniform distribution in magnitude range (0, 1) and in phase range $(0, \pi/2)$.

The filter designs and input sequences have then been passed to Right-Size, and for each design three different optimization procedures have been followed.

First, the design has been synthesized with the uniform scaling and the uniform word-length resulting in the optimum area estimate, given the error constraints. This design choice reflects the simplest form of optimized DSP design. Second, the design has been synthesized with scaling individually optimized for each signal (see Section 5.2) and the optimum uniform word-length. This design choice reflects the use of a tool such as [Stephenson et al. 2000] which focuses on optimizing signals from the MSB-side. The final design procedure has been to use an individually optimized scaling combined with an individually optimized word-length, as proposed by this paper.

From the filter designs in Simulink and the representative input sequences, Right-Size automatically generates a combination of structural VHDL and low-level integer arithmetic cores. In each case, this process took less than 10 minutes on a Pentium IV PC running Linux. Each design has been fully placed and routed in a Xilinx Virtex 1000 FPGA (XCV1000BG560-6), after which an area, power consumption, and timing analysis has been performed. Due to memory and run-time constraints imposed by large value-change-dump simulation files, power analysis could only be performed for hundreds of samples, rather than the thousands used by Right-Size, however it was observed that in practice after about 100 samples the power value reported stabilizes.

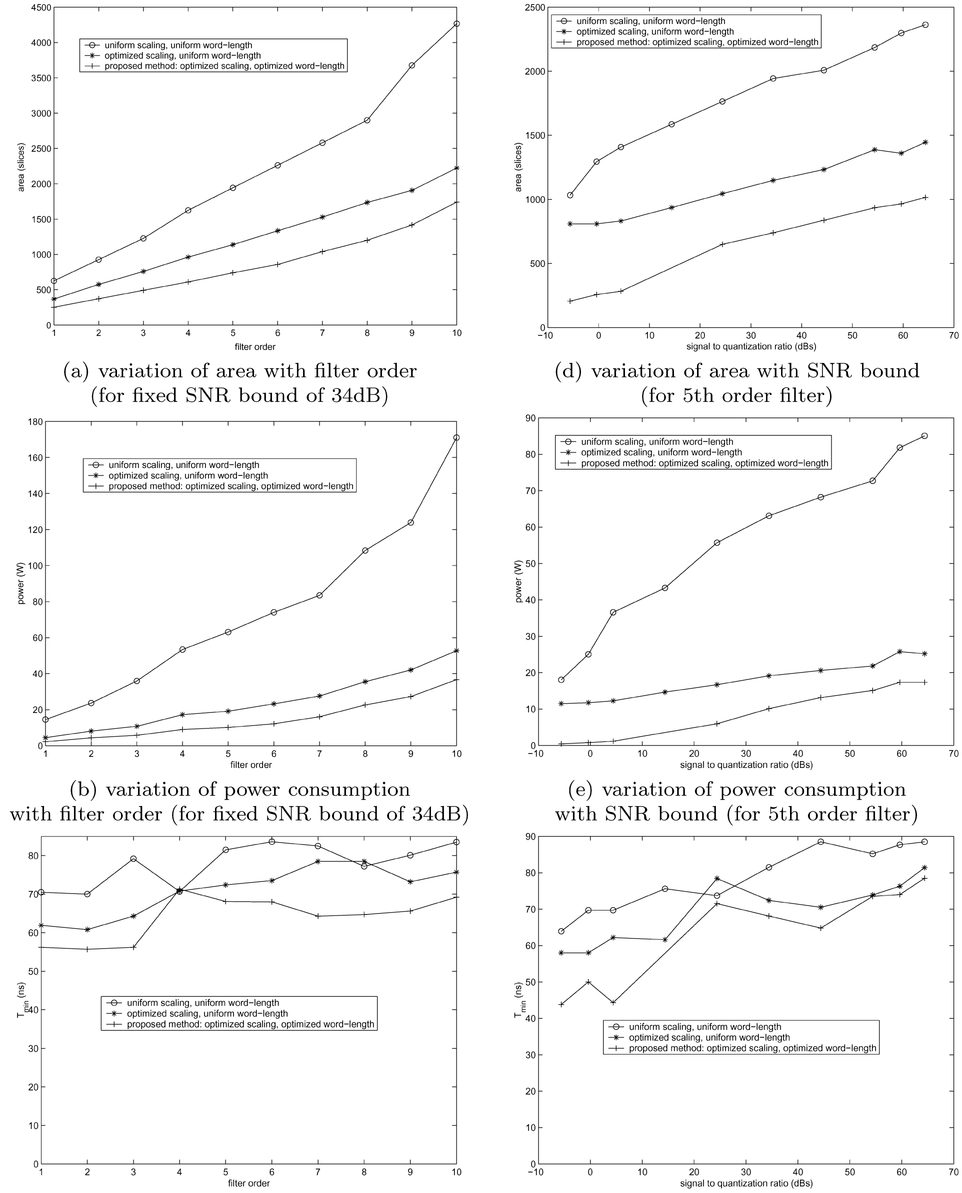
The first set of results is concerned with the variation of design metrics with the order of the filter to be synthesized. For each of these results, the filters have been synthesized using the same lower-bound on output SNR of 34dB, which was verified by simulation to have been achieved in all cases.

The results are illustrated in Figures 9(a), (b) and (c) for area, power, and clock period, respectively. Area savings of up to 37% (mean 32%) have been achieved over scaling optimization alone, and up to 63% (mean 61%) over neither scaling nor word-length optimization. This is *combined* with a power reduction of up to 49% (mean 43%) and speed-up of up to 18% (mean 10%) over scaling optimization alone, and a power reduction of up to 84.6% (mean 81.2%) and speed-up of up to 29% (mean 18%) over neither scaling nor word-length optimization.

The second set of results is concerned with the variation of design metrics with the user-specified lower-bound on allowable SNR. For these results, a 5th order LMS filter has been synthesized with SNR bound varying between −6dB and 64dB, again verified as achieved though simulation. These results are illustrated in Figures 9(d), (e) and (f) for area, power, and clock period, respectively. As well as demonstrating the useful capability to trade-off numerical accuracy for area, power and speed, these results also illustrate significant improvements in all three metrics.

Area savings have been achieved of up to 75% (mean 45%) over scaling optimization alone, and up to 80% (mean 66%) over neither scaling nor word-length optimization. This is *combined* with a power reduction of up to 96% (mean 58%) and speed-up of up to 29% (mean 11%) over scaling optimization alone, and a power reduction of up 98% (mean 87%) and speed up of up to 36% (mean 20%) over neither scaling nor word-length optimization.

It should be expected that on average the power savings are no smaller than the area savings of this approach. However, in practice, the power savings are



(a) variation of area with filter order (for fixed SNR bound of 34dB)

(d) variation of area with SNR bound (for 5th order filter)

(b) variation of power consumption with filter order (for fixed SNR bound of 34dB)

(e) variation of power consumption with SNR bound (for 5th order filter)

(c) variation of minimum realizable clock period with filter order (for fixed SNR bound of 34dB)

(f) variation of minimum realizable clock period with SNR bound (for 5th order filter)

Fig. 9. Synthesis results for LMS adaptive filters.

often significantly greater. This can be explained by two observations relating to the switching activity of signals. First, if the scaling of each signal is not individually optimized, then a significant number of signals will contain unnecessary sign-extension. When a two's complement signal changes from a positive to a negative value, or vice-versa, *all* of these MSBs will toggle. Thus, the overall

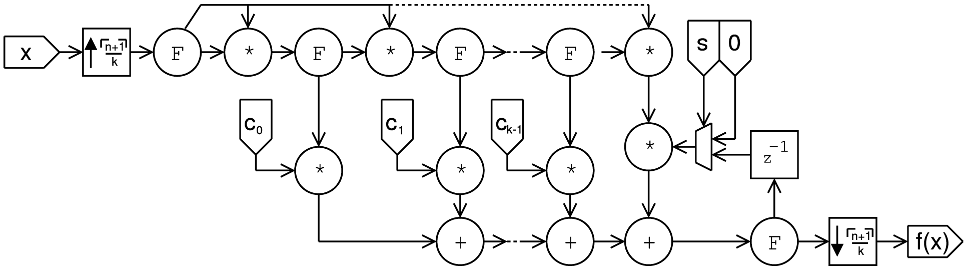


Fig. 10. General structure of a folded polynomial evaluation architecture (pipelining not shown).

switching activity in a realization can be reduced dramatically by applying scaling optimization [Mehendale and Sherlekar 2001]. Second, when a sampled signal is in a period of relatively low-frequency (with respect to the Nyquist rate [Mitra 1998]), the activity amongst low-order bits is significantly greater than that amongst high-order bits, due to the slowly changing signal value, a phenomenon that has been confirmed through simulation, and analysed in the context of Gaussian signals by Satyanarayana and Parhi [2000]. Thus, the proposed word-length optimization, which specifically targets the low-order bits of each signal, is likely to lead to a significant reduction in the overall activity level. In addition, through further simulation, it has been confirmed that large portion of the power consumption due to logic in these systems derives from multiplier cores. In multipliers, the power consumption is far more sensitive to reductions in the switching activity of low-order input bits than that of high-order input bits [Mehendale and Sherlekar 2001]. These explanations are supported by the plot of Figure 9(e) which shows the power saving of the proposed method over scaling optimization alone increasing rapidly for low SNR. This is because the low SNR allows word-length optimization to aggressively target more low-order bits.

7. CASE STUDY 2: POLYNOMIAL EVALUATION

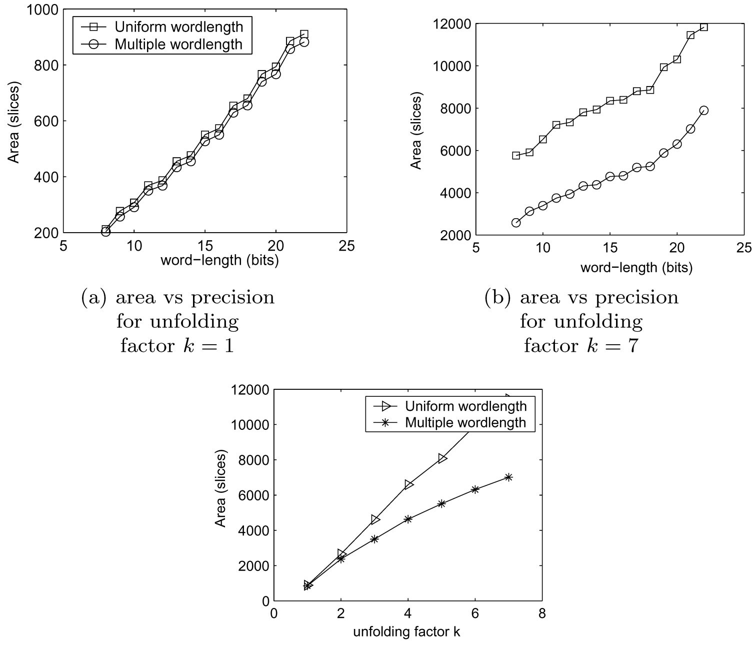
7.1 Folded Polynomial Evaluation: A Review

A diagram showing the general form of a folded polynomial evaluation architecture is shown in Figure 10. This architecture, adapted from [Sidahao et al. 2003], is capable of producing one nth order polynomial evaluation every $\lceil (n+1)/k \rceil$ cycles. Each ROM, labeled c_0 to c_{k-1} in Figure 10, contains $\lceil (n+1)/k \rceil$, entries where k is an unfolding factor.

Clearly, as k increases, the degree of *specialization* increases, as the number entries in each ROM is reduced. While for any folded architecture, the multipliers are general, rather than constant-coefficient multipliers, this increasing degree of specialization can have significant implications on reducing the number of "guard" bits [Muller 1997] required to achieve a faithfully rounded result.

7.2 Results

Figure 11(a) illustrates the change in area requirements for implementing a sine function as the precision of the required function increases, for a



(a) area vs precision for unfolding factor $k = 1$

(b) area vs precision for unfolding factor $k = 7$

(c) area vs unfolding factor k for 21-bit precision

Fig. 11. Area required for a sin function approximated by a 7th order polynomial.

fully-folded architecture, with a throughput of one evaluation every 8 cycles. Two plots are shown: one is the optimum uniform word-length implementation, and one is the optimized multiple word-length implementation resulting from the Right-Size tool (an optimized scaling is used in both cases). The latter consistently consumes less area than the former, but only by an average of 5%.

By contrast, Figure 11(b) illustrates the same plots for a significantly unfolded architecture: one with a throughput of one evaluation every 2 cycles. This time the difference between the two design approaches averages 50%.

This trend is brought out in Figure 11(c), where area is plotted against unfolding factor for a fixed precision. As the unfolding factor increases, the two curves separate. Usually the area of such a circuit is modelled as an approximately linear function of the unfolding factor, as the number of multipliers and adders grows linearly with unfolding factor. This is indeed a visible trend in the uniform word-length implementation. However, the multiple word-length area grows *sublinearly* in unfolding factor. This result can be explained, as the proposed procedure will automatically select the necessary scaling and word-length for each signal in order to maintain the required precision. The increasing specialization of the arithmetic components as the unfolding factor increases, results in significantly increased scope for optimization.

8. CONCLUSION

A novel error prediction procedure has been proposed for nonlinear systems containing differentiable nonlinearities. This procedure has been incorporated within the word-length optimization procedure first introduced in Constantinides et al. [2003], to produce a tool called Right-Size, which it is hoped will soon be released to the research community.

The power of the proposed technique has been illustrated by synthesizing several LMS adaptive filters and polynomial evaluation circuits. Results from these designs have shown area reductions averaging 66% combined with power reductions averaging 87% and speed-up averaging 20% over common alternative design strategies.

Word-length optimization has been shown to effectively take advantage of the differing switching activity in different bits of a two's complement signal, in order to reduce the overall power consumption.

Future work in this field will concentrate on providing a framework to approach non-differentiable nonlinearities such as threshold detectors, strongly nonlinear systems, and extending the approach to more control-intensive applications.

ACKNOWLEDGMENTS

The contributions of Mr. Abunaser Miah and Mr. Nalin Sidahao, and the guidance of Profs. Peter Y. K. Cheung and Wayne Luk are gratefully acknowledged.

REFERENCES

Aho, A. V., Sethi, R., and Ullman, J. D. 1986. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, Reading, MA.

Alippi, C. 2002. Randomized algorithms: A system-level, poly-time analysis of robust computation. *IEEE Trans. Comput. 51*, 7 (July), 740–749.

Cantin, M.-A., Savaria, Y., and Lavoie, P. 2001. An automatic word length determination method. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. IEEE Computer Society Press, Los Alamitos, CA. V-53–V-56.

Chang, M. L. and Hauck, S. 2004. Automated least-significant bit datapath optimization for FPGAs. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines* (Napa, CA). IEEE Computer Society Press, Los Alamitos, CA.

Cmar, R., Rijnders, L., Schaumont, P., Vernalde, S., and Bolsens, I. 1999. A methodology and design environment for DSP ASIC fixed point refinement. In *Proceedings of the Design Automation and Test in Europe* (Munich, 1999).

Constantinides, G. A. 2003. Perturbation analysis for word-length optimization. In *Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines* (Napa Valley, CA, Apr.). IEEE Computer Society Press, Los Alamitos, CA.

Constantinides, G. A., Cheung, P. Y. K., and Luk, W. 2003. Wordlength optimization for linear digital signal processing. *IEEE Trans. Computer-Aid. Des. Integ. Circ. Syst. 22*, 10 (Oct.), 1432–1442.

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

Constantinides, G. A. and Woeginger, G. J. 2002. The complexity of multiple wordlength assignment. *Appl. Math. Lett. 15*, 2 (Feb.), 137–140.

Fang, C. F., Rutenbar, R. A., Püschel, M., and Chen, T. 2003. Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling. In *Proceedings of the ACM/IEEE Design Automation Conference* (Anaheim, CA). ACM, New York, 496–501.

FREETEL. 1993. Esprit project 6166: FREETEL database.

Gaffar, A., Mencer, O., Luk, W., and Cheung, P. 2004. Unifying bit-width optimisation for fixed-point and floating-point designs. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines* (Napa, CA). IEEE Computer Society Press, Los Alamitos, CA.

HAYKIN, S. S. 1996. *Adaptive Filter Theory*. Prentice Hall, Englewood Cliffs, NJ.

HWANG, J., MILNE, B., SHIRAZI, N., AND STROOMER, J. 2001. System level tools for DSP in FPGAs. In *Proceedings of the Field Programmable Logic*, R. Woods and G. Brebner, Eds. (Berlin, Germany). Springer-Verlag, New York.

JACKSON, L. B. 1970. On the interaction of roundoff noise and dynamic range in digital filters. *Bell Syst. Tech. J. 49*, (Feb.), 159–184.

KEDING, H., WILLEMS, M., COORS, M., AND MEYR, H. 1998. FRIDGE: A fixed-point design and simulation environment. In *Proceedings of the Design Automatation and Test in Europe* (1998).

KUM, K.-I. AND SUNG, W. 2001. Combined word-length optimization and high-level synthesis of digital signal processing systems. *IEEE Trans. Computer Aid Des. 20*, 8 (Aug.), 921–930.

LEE, E. A. AND MESSERSCHMITT, D. G. 1987. Static scheduling of synchronous dataflow programs for digital signal processing. *IEEE Trans. Computers 36*, 1 (Jan.), 24–35.

MEHENDALE, M. AND SHERLEKAR, S. D. 2001. *VLSI Synthesis of DSP Kernels*. Kluwer, Dordrecht, Netherlands.

MITRA, S. K. 1998. *Digital Signal Processing*. McGraw-Hill, New York.

MULLER, J. 1997. *Elementary Functions*. Springer-Verlag, Berlin, Germany.

NAYAK, A., HALDAR, M., CHOUDHARY, A., AND BANERJEE, P. 2001. Precision and error analysis of MATLAB applications during automated hardware synthesis for FPGAs. In *Proceedings of the Design Automation and Test in Europe* (Munich, Germany). 722–728.

SATYANARAYANA, J. AND PARHI, K. 2000. Theoretical analysis of word-level switching activity in the presence of glitching and correlation. *IEEE Trans. VLSI Syst. 8*, 2, 148–159.

SEDRA, A. S. AND SMITH, K. C. 1991. *Microelectronic Circuits*. Saunders, New York.

SIDAHAO, N., CONSTANTINIDES, G. A., AND CHEUNG, P. Y. K. 2003. Architectures for function evaluation on FPGAs. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. IEEE Computer Society Press, Los Alamitos, CA.

SIMULINK. Simulink. http://www.mathworks.com.

STEPHENSON, M., BABB, J., AND AMARASINGHE, S. 2000. Bitwidth analysis with application to silicon compilation. In *Proceedings of the SIGPLAN Programming Language Design and Implementation* (Vancouver, British Columbia, June). AG, New York.

WADEKAR, S. A. AND PARKER, A. C. 1998. Accuracy sensitive word-length selection for algorithm optimization. In *Proceedings of the International Conference on Computer Design* (Austin, Texas, Oct.). 54–61.

XILINX, INC. 2002. *Field Programmable Gate Arrays*. Xilinx, Inc., San Jose, CA.

Received May 2004; revised March 2005 and July 2005; accepted July 2005