Perturbation Analysis for Word-length Optimization

George A. Constantinides
Department of Electrical and Electronic Engineering, Imperial College, London SW7 2BT, U.K.
george.constantinides@ieee.org

Abstract

This paper introduces a design tool and its associated procedures 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. Also in this paper, the power-optimizing capabilities of word-length optimization are studied for the first time. Application of the proposed procedure to adaptive filters realized in a Xilinx Virtex FPGA has resulted in 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.*

1 Introduction

This paper is concerned with design automation for digital signal processing (DSP) algorithms implemented on field-programmable gate arrays (FPGAs). 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. 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 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.

A tool called Right-Size has been implemented, which performs the proposed procedures and automates the design flow. The software is designed in a highly modular, extensible object-oriented manner, allowing any building block operations to be easily incorporated. It is hoped to soon release this tool to the research community – interested parties are encouraged to contact the author.

The contributions of this paper 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 [1, 2], which was limited to linear systems
- the first 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 introduction to design using the Right-Size tool, which implements the proposed optimization procedures

The rest of this paper 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, and section 6 examines its application to a particular class of signal processing algorithms: LMS adaptive filters. Results from these filters are discussed in section 6.2, after which conclusions are drawn in section 7.

2 Background

Several sections of this paper will refer to the concepts of *linearity* and *time-invariance*. A system exhibiting the

Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'03)
1082-3409/03 $17.00 © 2003 IEEE

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. For more precise definitions and further information, any introductory signal processing text such as [3] may be consulted.

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 [4, 5].

In [6] 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 [1, 7, 8] or do not support different fractional precision for different internal variables [9]. An optimum approach has also been proposed [2], which is limited to linear time-invariant systems.

Benedetti and Perona [10] have proposed an analytic method for word-length optimization of general systems based on interval arithmetic. The authors concentrate on error-free word-length determination in loop-free data-flow graphs, and thus do not consider the problem of allowing a trade-off between accuracy and other figures of merit.

The Bitwise Project [11] proposes a similar compiler-based technique based on propagating the ranges of integer variables backwards and forwards through integer data-flow graphs. The focus is on removing unwanted most-significant bits (MSBs), and no technique is proposed for removing unwanted least-significant bits (LSBs). The authors demonstrate area savings of 15% to 86% combined with speed increases of up to 65% over the use of 32-bit integers for all variables.

The MATCH Project [9] also uses compiler-based propagation through data-flow graphs, except variables with a fractional component are allowed. All signals in the model must have equal fractional precision; the authors propose an analytic worst-case error model in order to estimate the required number of fractional bits, applicable to loop-free data-flow graphs. Area reductions of 80% combined with speed increases of 20% are reported when compared to a uniform 32-bit representation. Recently this design flow has been updated [12] to support a combination of simulation and analysis.

Wadekar and Parker [7] have also proposed a methodology for word-length optimization. Like [9], this technique allows controlled worst-case error at system outputs, however each intermediate variable is allowed a distinct word-length. Series expansions are used to evaluate an estimate of the worst-case error at the output of a loop-free data-flow graph. Results indicate area reductions of between 15% and 40% over the optimum uniform word-length implementation.

Cmar *et al.* [13] have developed a word-length optimization system which uses a combination of range propagation and simulation with known input vectors to limit the word-lengths of internal variables. No mechanism is proposed to automate the trade-off of system area against error.

Kum and Sung [8] have proposed several word-length optimization techniques to trade-off system area against system error, some of which have been incorporated in the Cadence Signal Processing Worksystem. These techniques are heuristics based on bit-true simulation of the design under various internal word-lengths. Indeed, simulation to determine error properties has recently been given a firm theoretical footing by Alippi [14].

In previous work [1, 2], 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.

In summary, several systems for word-length determination have been proposed in the past. Some use an analytic approach to scaling and/or error estimation [1, 2, 7, 9, 10, 11], some use simulation [8], and some use a hybrid of the two [12, 13]. Analytic techniques often have the advantage of speed over simulation-based approaches, however they tend to be less general (applicable only to linear systems [1, 2] or only to systems with loop-free data-flow graphs [7, 9, 10, 11]).

3 Design Flow

Before discussing the internal details of the 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 [15]. Simulink is widely used within the DSP community, and has been recently incorporated

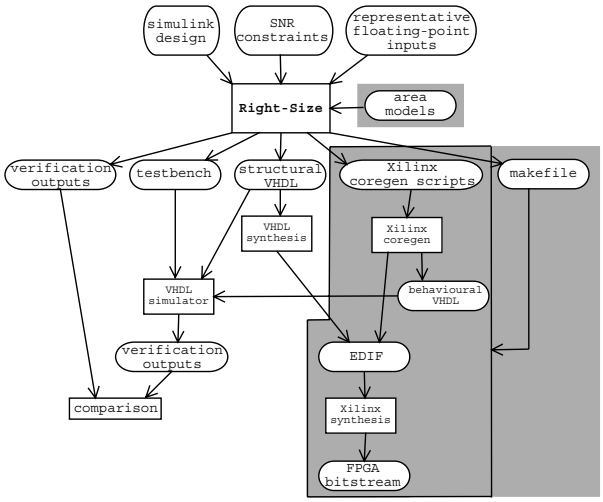


Figure 1: Design Flow (shaded portions are FPGA vendor-specific)

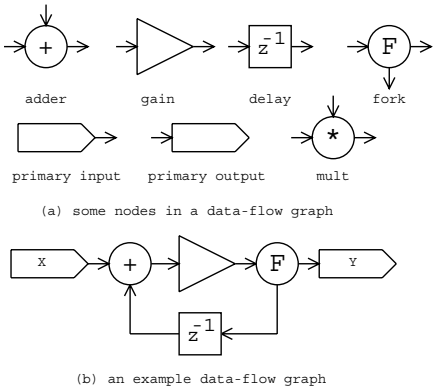


Figure 2: The graphical representation of a data-flow graph

into the Xilinx System Generator [16] and Altera DSP builder design flows. Our tool can either use its own textual input format to describe a DSP system, or can read directly from Simulink models providing a fast design route for DSP engineers. At the design stage, the engineer need not specify any hardware-specific signal features such as their word-length or scaling (binary point location). 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 design tool thus represents a truly 'behavioural' 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 Fig. 1, with the vendor-specific portions shaded. The proposed technique is entirely architecture independent, although the exact savings possible by using the approach may vary from architecture to architecture. In order to re-target the synthesis tool to a different architecture 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 Fig. 1, the inputs to the Right-Size system are a specification of the system behaviour (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 behavioural 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 post-Right-Size synthesis process.

4 Algorithm Representation

Individual computations in a design can be one of many types. The described procedure requires each of these computation types to produce a differentiable function of its inputs. The types currently implemented in Right-Size and discussed in this paper are: constant coefficient multiplication, general multiplication, unit-sample delay, addition, and branching (fork). Only fork nodes are allowed to have greater than unit outdegree. It is straight-forward to extend Right-Size to other node types using the class structure available.

The representation used as a starting point for the optimization techniques described in this paper is referred to as a data-flow graph (DFG) [17]. 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 paper, DFGs will be visualized using a graphical representation, as shown in Fig. 2.

5 Proposed Procedure

The proposed procedure consists of three steps. Firstly a perturbation analysis is performed, in order to quantify the sensitivity of each system output to noise at each point within the computational structure. Secondly 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 which 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 [1] and shown to produce near-optimal results for small systems [2].

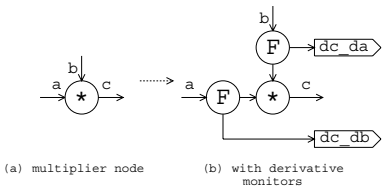


Figure 3: Local graph transformation to insert derivative monitors

5.1 Perturbation Analysis

In order to make some of the analytical results on error sensitivity for linear, time-invariant systems [1] 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 behaviour of the system. Under such circumstances, each component in the system can be locally linearized, or replaced by its "small-signal equivalent" [18] in order to determine the output behaviour under a given rounding scheme.

We shall 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 $y[t] \approx x_1[t]\frac{\partial f}{\partial X_1} + \ldots + x_n[t]\frac{\partial f}{\partial X_n}$.

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, we have produced a linear time-varying small-signal model for a nonlinear time-invariant component.

The linearity of the resulting model allows us to predict the error at system outputs due to *any* small perturbation of signal $s \in S$ analytically, given the simulation-obtained error of a *single* perturbation instance 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, Right-Size takes advantage of the static schedulability of the synchronous data-flow [17] model implied by the algorithm representation, leading to an exceptionally fast simulation compared to event-driven simulation.

5.1.1 Derivative Monitors

In order to construct the small-signal model, Right-Size must first evaluate the differential coefficients of the Taylor series model for nonlinear components. Like other procedures described in this paper, this 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 differential. The general multiplier is the only nonlinear component considered in the present paper; the graph transformation for multipliers is illustrated in Fig. 3. Since $f(X_1, X_2) = X_1X_2$, $\frac{\partial f}{\partial X_1} = X_2$ and $\frac{\partial f}{\partial X_2} = X_1$.

After insertion of the monitors, the simulation method is called to provide a (double precision floating point) simulation of the entire system, and write-out the derivatives to appropriate data files to be used by the linearization process, to be described below.

5.1.2 Linearization

The construction of the small-signal model may now proceed, again through graph transformation. All linear components (adder, constant-coefficient multiplier, fork, delay, primary input, primary output) remain unchanged as a result of the linearization process. Each nonlinear component is replaced by its Taylor model. Additional primary inputs are added to the DFG to read the Taylor coefficients from the derivative monitor files created by the above large-signal simulation.

As an example, the Taylor expansion transformation for the multiplier node is illustrated in Fig. 4. Note that the graph portion in Fig. 4(b) still contains multiplier 'nonlinear' components, although one input of each multiplier node is now external to the model. This absence of feedback ensures linearity, although not time-invariance.

5.1.3 Noise Injection

The SNR measure of signal quality is used to specify the user-acceptable limits on signal distortion through truncation or rounding. In [1], 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

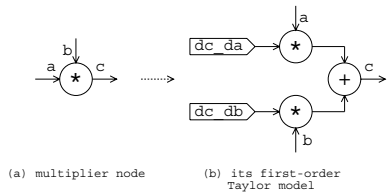


Figure 4: Local graph transformation to produce small-signal model

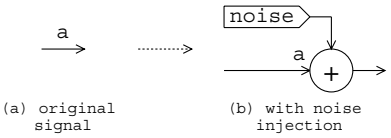


Figure 5: Local graph transformation to inject perturbations

injected at each point of quantization. Such a purely analytic technique can be used only for linear time-invariant systems, however in this paper we suggest an extension of the approach for nonlinear systems.

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 can be determined *once only* through simulation, this response can be scaled with analytically derived coefficients in order to determine the response to *any* rounding or truncation scheme.

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. In our case, 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 Fig. 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 [19] which leads to significantly faster simulation results for nontrivial DFGs.

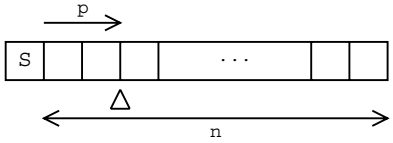


Figure 7: A multiple word-length signal ('S' represents the sign bit), the triangle represents the binary-point location

The entire process is illustrated for a simple DFG in Fig. 6. The original DFG is illustrated in Fig. 6(a). The perturbation analysis will be performed for the signals marked (*) and (**) in this figure. After inserting derivative monitors for nonlinear components, the transformed DFG is shown in Fig. 6(b). The linearized DFG is shown in Fig. 6(c), and its two variants for the signals (*) and (**) are illustrated in Figs. 6(d) and (e) respectively. Finally, the corresponding simplified DFGs after zero-propagation are shown in Figs. 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 [1] and illustrated in Fig. 7.

Scaling analysis is performed in two phases. Firstly a simulation is performed using the user-provided input sequence. The peak signal value P_s reached by 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. 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, i.e. $\hat{p}_s < p_s$, then p_s is set equal to $\hat{p}_s$. Such a conditioning [1] 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 RightSize is identical to that described for linear time-invariant systems in [1].

By using the perturbation sensitivities calculated as described in section 5.1, estimation of the error at system out-

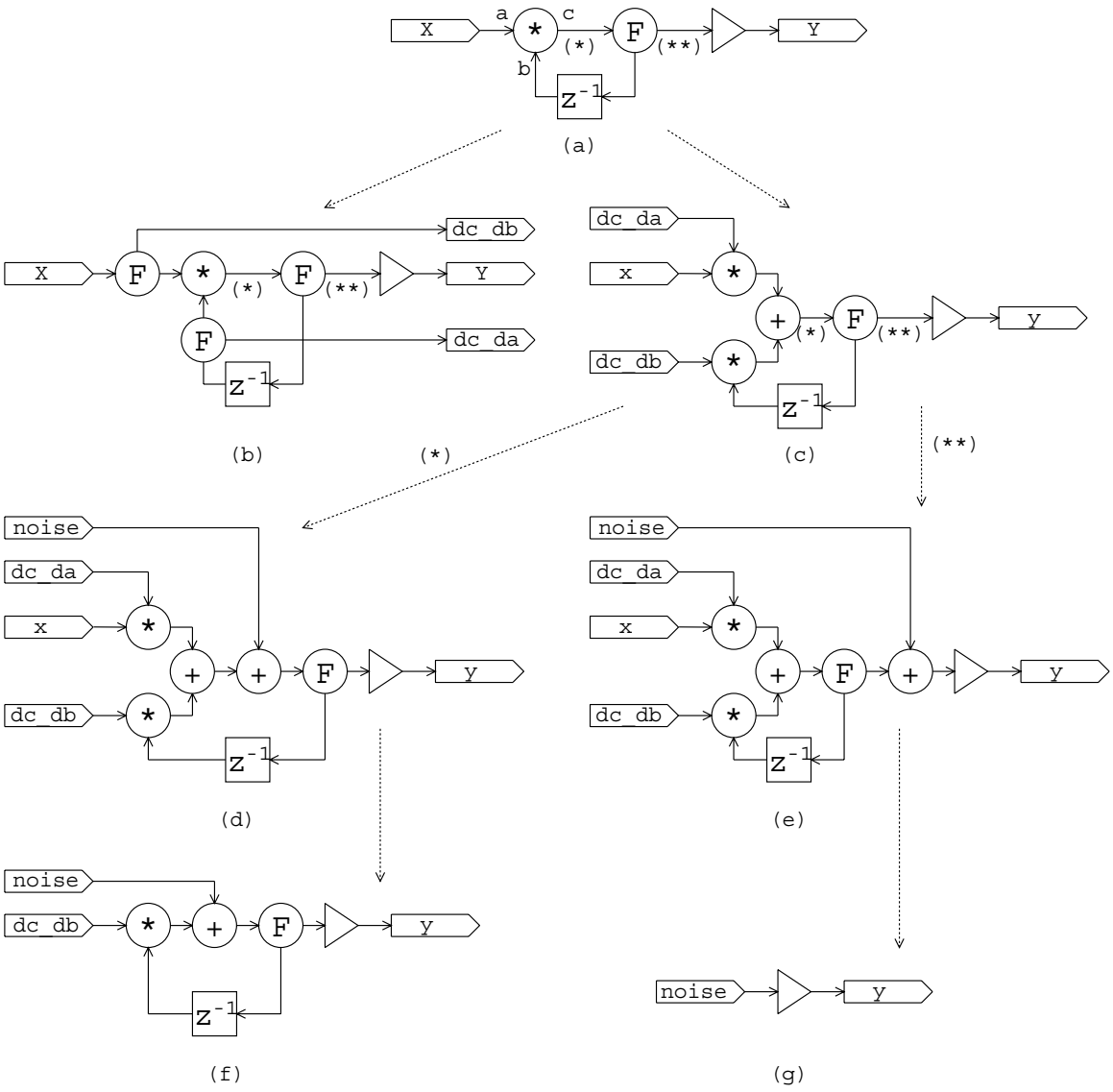


Figure 6: Example perturbation analysis

puts, 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 [1]. 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 FPGA architecture. Currently, Right-Size targets Xilinx Virtex FPGAs [20], 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: 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 multi-path distortion in mobile communication systems [21].

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 [7, 9, 10, 11, 13] 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 non-linear systems, limiting the applicability of purely analytic techniques such as [1, 2].
- 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 so-called least-mean-square (LMS) adaptive filter [21] will be considered in this section, due to its widespread use in practice. For the unfamiliar reader, a brief review of LMS filters will now be provided.

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 $\mathbf{u}[t] = (x[t]\ x[t-1]\ x[t-2]\ \ldots\ x[t-n])^T$, where ^{T} represents vector/matrix 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 μ is a user-chosen scalar adaptation coefficient.

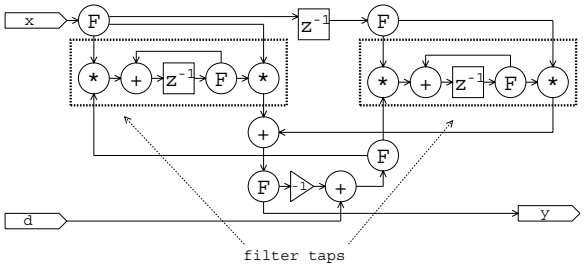


Figure 8: First order LMS adaptive filter

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

A DFG for a first-order LMS filter is shown in Fig. 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 in Simulink and passed to Right-Size for synthesis. In each case the 'desired' input $d[t]$ to the adaptive filter is a well-known 100,000 sample voice clip from [22]. 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. Firstly, the design has been synthesized with the optimum uniform

scaling and the optimum uniform word-length for all signals. This design choice reflects the simplest form of optimized DSP design. Secondly, 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 [11] 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 Xilinx Coregen scripts, together with a makefile to synthesize the Virtex bit-stream. Each design has been fully placed and routed in a Xilinx Virtex 1000 (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 a 100-sample portion of the 100,000-sample input sequence used by Right-Size.

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.

The results are illustrated in Fig. 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. These results are illustrated in Fig. 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 often significantly greater. This can be explained by two observations relating to the switching activity of signals. Firstly, 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 switching activity in a realization can be reduced dramatically by applying scaling optimization. Secondly, when a sampled signal is in a period of relatively low-frequency (with respect to the Nyquist rate [3]), the activity amongst low-order bits is, on average, likely to be significantly larger than that amongst high-order bits due to the slowly changing signal value. Thus 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, it is likely that a large portion of the power consumption due to logic activity in DSP 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 [23]. These explanations are supported by the plot of Fig. 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.

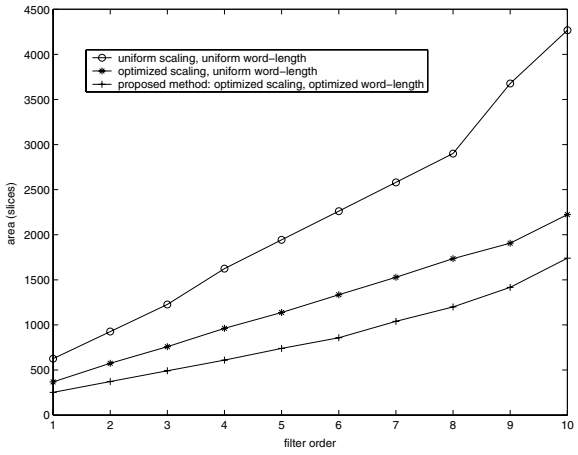
7 Conclusion

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 [1], to produce a tool called Right-Size, which it is hoped will soon be released to the research community.

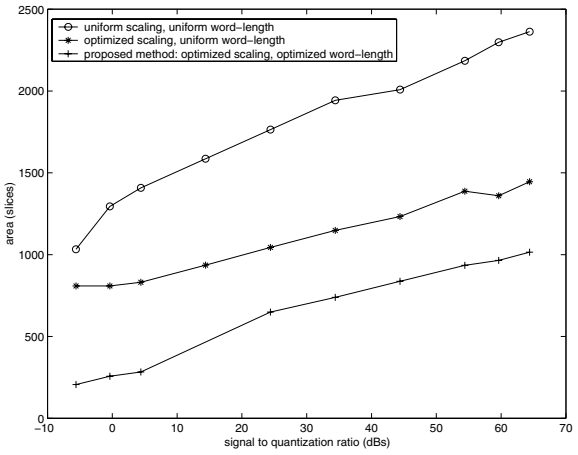
The power of the proposed technique has been illustrated by synthesizing several LMS adaptive filters. Results from these filters have shown area reductions of up to 80% combined with power reductions of up to 98% and speed-up of up to 36% over common alternative design strategies.

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.

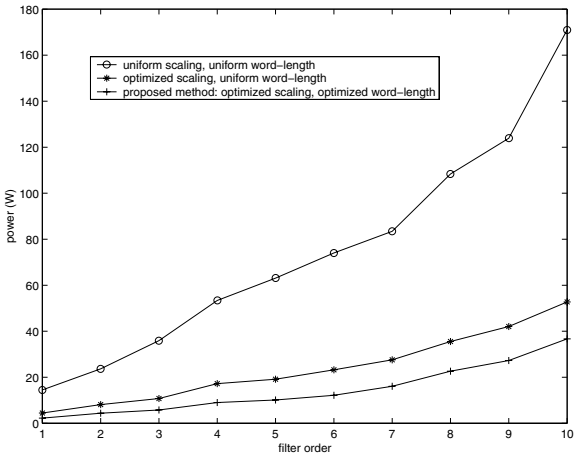
Future work in this field will concentrate on extending the range of operations known to the Right-Size tool, and providing a framework to approach non-differentiable nonlinearities such as threshold detectors.



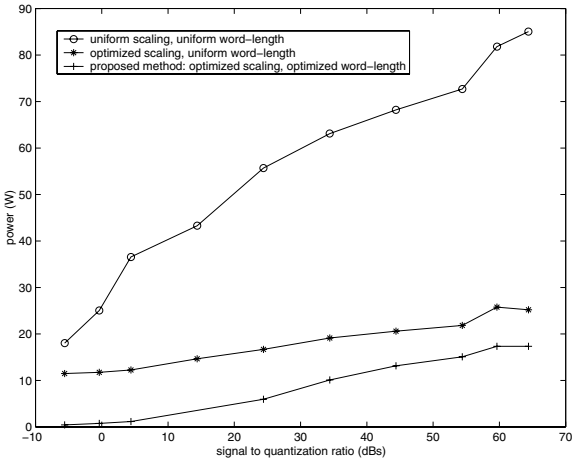
(a) variation of area with filter order (for fixed SNR bound of 34dB)



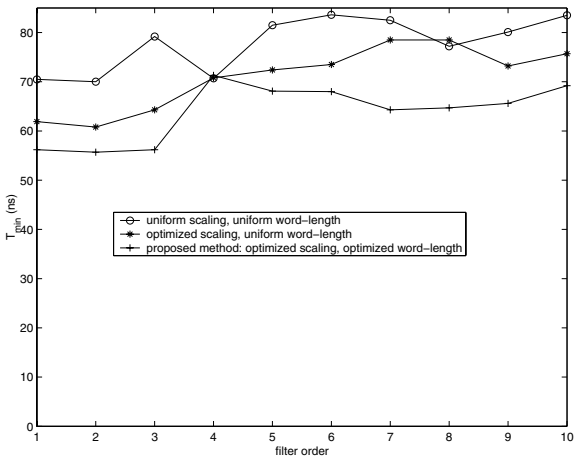
(d) variation of area with SNR bound (for 5th order filter)



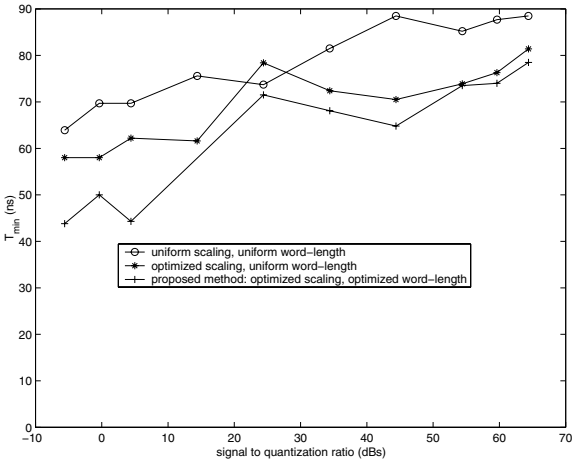
(b) variation of power consumption with filter order (for fixed SNR bound of 34dB)



(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)

Figure 9: Synthesis results for LMS adaptive filters

Acknowledgements

The guidance of Prof. Peter Cheung and Dr. Wayne Luk are gratefully acknowledged.

References

[1] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "The multiple wordlength paradigm," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, Rohnert Park, CA, April–May 2001.

[2] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Optimum wordlength allocation," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 2002.

[3] S. K. Mitra, *Digital Signal Processing*, McGraw-Hill, New York, 1998.

[4] A. A. Gaffar, W. Luk, P. Y. K. Cheung, N. Shirazi, and J. Hwang, "Automating customization of floating-point designs," in *Proc. Field-Programmable Logic and Applications*. 2002, pp. 523–533, Springer-Verlag.

[5] A. A. Gaffar, O. Mencer, W. Luk, P. Y. K. Cheung, and N. Shirazi, "Floating-point bitwidth analysis via automatic differentiation," in *Proc. IEEE International Conference on Field-Programmable Technology*, Hong Kong, 2002.

[6] G. A. Constantinides and G. J. Woeginger, "The complexity of multiple wordlength assignment," *Applied Mathematics Letters*, vol. 15, no. 2, pp. 137–140, February 2002.

[7] S. A. Wadekar and A. C. Parker, "Accuracy sensitive word-length selection for algorithm optimization," in *Proc. International Conference on Computer Design*, Austin, Texas, October 1998, pp. 54–61.

[8] K.-I. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Trans. Computer Aided Design*, vol. 20, no. 8, pp. 921–930, August 2001.

[9] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee, "Precision and error analysis of MATLAB applications during automated hardware synthesis for FPGAs," in *Proc. Design Automation and Test in Europe*, Munich, Germany, 2001, pp. 722–728.

[10] A. Benedetti and P. Perona, "Bit-width optimization for configurable DSP's by multi-interval analysis," in *Proc. 34th Asilomar Conference on Signals, Systems and Computers*, 2000.

[11] M. Stephenson, J. Babb, and S. Amarasinghe, "Bitwidth analysis with application to silicon compilation," in *Proc. SIGPLAN Programming Language Design and Implementation*, Vancouver, British Columbia, June 2000.

[12] M. L. Chang and S. Hauck, "Précis: A design-time precision analysis tool," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, 2002.

[13] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens, "A methodology and design environment for DSP ASIC fixed point refinement," in *Proc. Design Automation and Test in Europe*, München, 1999.

[14] C. Alippi, "Randomized algorithms: A system-level, poly-time analysis of robust computation," *IEEE Trans. on Computers*, vol. 51, no. 7, pp. 740–749, July 2002.

[15] "Simulink," http://www.mathworks.com.

[16] J. Hwang, B. Milne, N. Shirazi, and J. Stroomer, "System level tools for DSP in FPGAs," in *Proc. Field Programmable Logic*, R. Woods and G. Brebner, Eds. 2001, Springer-Verlag.

[17] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous dataflow programs for digital signal processing," *IEEE Trans. Computers*, January 1987.

[18] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, Saunders, 1991.

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

[20] Xilinx, Inc., San Jose, *Field Programmable Gate Arrays*, 1998.

[21] S. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

[22] FREETEL, "Esprit project 6166: FREETEL database," 1993.

[23] M. Mehendale and S. D. Sherlekar, *VLSI Synthesis of DSP Kernels*, Kluwer, 2001.