Wordlength Optimization for Linear Digital Signal Processing

George A. Constantinides, Peter Y. K. Cheung, and Wayne Luk

***Abstract*—This paper presents an approach to the wordlength allocation and optimization problem for linear digital signal processing systems implemented as custom parallel processing units. Two techniques are proposed, one which guarantees an optimum set of wordlengths for each internal variable, and one which is a heuristic approach. Both techniques allow the user to tradeoff implementation area for arithmetic error at system outputs. Optimality (with respect to the area and error estimates) is guaranteed through modeling as a mixed integer linear program. It is demonstrated that the proposed heuristic leads to area improvements of 6% to 45% combined with speed increases compared to the optimum uniform wordlength design. In addition, the heuristic reaches within 0.7% of the optimum multiple wordlength area over a range of benchmark problems.**

***Index Terms*—Bitwidth, digital signal processor (DSP), optimization, precision, wordlength.**

I. Introduction

This paper explores and examines design automation for linear time-invariant (LTI) applications. A design paradigm is proposed based on nonuniformity of signal scaling and precision between physically distinct parallel processing elements. It is shown how the fixed-point quantization effects of such designs can be estimated analytically, and that the design process can be automated from an initial infinite-precision behavioral specification.

The synthesis techniques presented in this paper are *lossy synthesis* approaches [1]. This term is used to denote that, while the behavior of the synthesized system is not identical to that of the algorithm specification (due to fixed-point quantization effects), the error can be specified and controlled by the user of the synthesis system, to enable tradeoffs with other figures of merit. Both optimum (with respect to the area and error estimates) and heuristic approaches to wordlength determination are proposed. The heuristic technique presented results in area improvements from 6% to 45% combined with speed increases when applied to standard signal processing benchmarks, while coming within 0.7% of the optimum area over small benchmark problems.

The digital signal processing (DSP) systems targeted by the optimization technique discussed in this paper are LTI systems [2]. This class of systems includes the most widely used DSP operations: finite-impulse response (FIR) and infinite-impulse response (IIR) digital filters, as well as linear transformations such as the discrete cosine transform, Fourier transform, and RGB to YCrCb conversion. Although not every DSP system is LTI, important subsystems will typically have this property.

Section II summarizes the relevant background, after which the architectures and models used are introduced in Section III, ending in Section III-G with a statement of the problem addressed in this paper. In Section IV, it is shown how this problem can be transformed into a mixed-integer linear program (MILP), and thus be solved using standard solvers. A heuristic technique is introduced in Section V.

The ideas presented have been implemented within the Synoptix high-level synthesis system [1] which, given a Simulink block diagram and user-specified bounds on roundoff noise for each of the outputs, creates an area-optimized DSP algorithm implementation in a structural hardware description language suitable for implementation on field-programmable gate arrays (FPGAs). Results from the Synoptix system are presented in Section VI before concluding the paper in Section VII.

The main original contributions of this paper are, therefore, a detailed exposition of the multiple wordlength design paradigm, including an approach which captures the full nonconvexity of the design space [3] and techniques for optimal and heuristic lossy synthesis of linear time invariant systems. Specific features of this analysis include

- formulation of the multiple wordlength assignment problem;
- a novel transformation of this problem into a mixed integer linear program;
- a novel heuristic for the optimization of wordlengths in a multiple wordlength system;
- an evaluation of the proposed heuristic in terms of its impact on system area and speed, compared to the optimum uniform wordlength implementation;
- a comparison of solution quality between the proposed heuristic and the optimum results.

II. Background

In [4], it has been demonstrated that a simplified version of the problem addressed in this paper is NP-hard. There are, however, several published approaches to wordlength optimization. Those offering an area/signal quality tradeoff are of a heuristic nature [1], [5], [6] do not support different fractional precision for different internal variables [7], or assume that arithmetic error falls monotonically in each signal wordlength [6], [8]. In contrast, this paper proposes techniques which capture the full complexity of the error constraint surface, allow multiple fractional precisions, and provide the choice of a heuristic or an optimum solution (with respect to the area and error models).

Some published approaches to the wordlength optimization problem use an analytic approach to scaling and/or error estimation [5], [7], [9], some use simulation [6], [8], and some use a hybrid of the two [10]. 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. There is little analytical work on supporting dataflow graphs containing cycles, although in [9], finite loop bounds are supported. Some published approaches use worst-case instantaneous error as a measure of signal quality [5], [7], [8] whereas some use signal-to-noise ratio (SNR) [1], [6]. The proposals in this paper use fully analytic error models, use SNR as a quality metric, and can be applied to dataflow graphs containing cycles without the pessimism exhibited by other analytic work. This is achieved at the cost of limiting the domain of application to LTI systems.

The remainder of this section reviews in more detail some of the more important literature in the field.

The Bitwise Project [9] proposes propagation of integer variable ranges backward and forward through dataflow graphs. The focus is

Manuscript received November 18, 2002; revised March 14, 2003. This work was supported in part by Hewlett-Packard Laboratories and in part by the Engineering and Physical Sciences Research Council, U.K. This paper was recommended by Associate Editor R. Gupta.

G. A. Constantinides and P. Y. K. Cheung are with the Department of Electrical and Electronic Engineering, Imperial College, London SW7 2BT, U.K. (e-mail: g.constantinides@ic.ac.uk; p.cheung@ic.ac.uk).

W. Luk is with the Department of Computing, Imperial College, London SW7 2BZ, U.K. (e-mail: wl@doc.ic.ac.uk).

Digital Object Identifier 10.1109/TCAD.2003.818119

0278-0070/03$17.00 © 2003 IEEE

on removing unwanted most-significant bits (MSBs). Results from integration in a synthesis flow indicate that area savings of between 15% and 86% combined with speed increases of up to 65% can be achieved compared to using 32-bit integers for all variables.

The MATCH Project [7] also uses range propagation through dataflow graphs, except variables with a fractional component are allowed. All signals in the model of [7] must have equal fractional precision; the authors propose an analytic worst-case error model in order to estimate the required number of fractional bits. Area reductions of 80% combined with speed increases of 20% are reported when compared to a uniform 32-bit representation.

Wadekar and Parker [5] have also proposed a methodology for wordlength optimization. Like [7], this technique also allows controlled worst-case error at system outputs, however, each intermediate variable is allowed to take a wordlength appropriate to the sensitivity of the output errors to quantization errors on that particular variable. Results indicate area reductions of between 15% and 40% over the optimum uniform wordlength implementation.

Kum and Sung [6] and Cantin *et al.* [8] have proposed several wordlength optimization techniques to tradeoff system area against system error. These techniques are heuristics based on bit-true simulation of the design under various internal wordlengths.

A useful survey of algorithmic procedures for wordlength determination has been provided by Cantin *et al.* [11]. In this work, existing heuristics are classified under various categories. However, the "exhaustive" and "branch-and-bound" procedures described in [11] do not necessarily capture the optimum solution to the wordlength determination problem, due to nonconvexity in the constraint space: it is actually possible to have a *lower* error at a system output by reducing the wordlength at an internal node [3]. Such an effect is not modeled in the surveyed articles, and forms one of the novel elements of the MILP model proposed in this paper. The novelty of the proposed heuristic lies in the use of the error constraint, rather than just the objective function, to guide the optimization procedure. As a result, the heuristic proposed in this paper does not fall into one of the categories described in [11].

III. Architectures and Models

A. Notation

For a directed graph $G(V, E)$ with node set V and edge set $E \subseteq V \times V$, $\mathrm{od}(v)$ denotes the outdegree of a node $v \in V$, $\mathrm{in}(v)$ denotes the set of in-edges incident to node v and $\mathrm{out}(v)$ denotes the set of out-edges incident to node v.

Set subtraction is indicated by the operator $\setminus$, $\mathbb{Z}$ is used to represent the set of integers, $\mathbb{N}$ is used to represent the set of positive integers, and $\mathbb{R}$ is used to represent the set of real numbers.

For a function $f : X \rightarrow Y$, $f(X' \subseteq X) \subseteq Y$ denotes the subset $\{y \in Y | \exists x \in X' : f(x) = y\}$.

$\lfloor x \rfloor$ represents the largest integer less than or equal to x. $\wedge$ is used to represent logical conjunction and $L_2\{H(z)\}$ represents the L_2 norm of a z-domain transfer function $H(z)$[2].

B. Algorithm Representation

Definition 1: A *computation graph* $G(V, S)$ 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*. Only FORK nodes, representing branching dataflow, are allowed to have greater than unit outdegree.

The node set V may be partitioned into subsets with the same type $V = V_I \cup V_O \cup V_A \cup V_D \cup V_G \cup V_F$ representing the INPORT, OUTPORT, ADD, DELAY, GAIN, and FORK nodes, respectively.

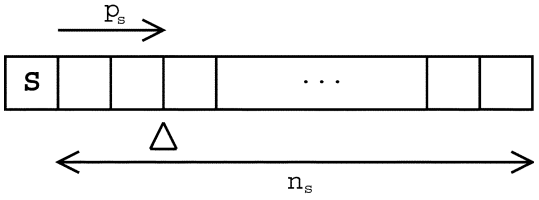

Fig. 1. Multiple wordlength paradigm (**S** indicates sign bit, n_s denotes wordlength and p_s denotes scaling).

If $V_G \subset V$ denotes the subset of nodes of GAIN type, then $cw : V_G \rightarrow \mathbb{N}$ is a function mapping the GAIN node to its coefficient wordlength, and $sc : V_G \rightarrow \mathbb{Z}$ is a function mapping the gain node to its scaling or binary point location.

Note that this paper only considers the problem of optimizing signal wordlengths; coefficient wordlengths are predefined and form part of the design specification. This is common practice in LTI DSP design, as coefficient quantization changes the system transfer functions only, and can thus be considered within the framework of traditional linear system theory.

Definition 1 is sufficiently general to allow any multiple input, multiple output (MIMO) LTI system to be modeled.

C. Multiple Wordlength Paradigm

Each two's-complement signal $s \in S$ in a multiple wordlength implementation of computation graph $G(V, S)$, has two parameters n_s and p_s, as illustrated in Fig. 1. The parameter n_s represents the number of bits in the representation of the signal (excluding the sign bit), and the parameter p_s represents the displacement of the binary point from the least-significant bit (LSB) side of the sign bit toward the LSB. Note that there are no restrictions on p_s; the binary point could lie outside the number representation, i.e., $p_s < 0$ or $p_s > n_s$.

This multiple wordlength implementation style inherits the speed, area, and power advantages of traditional fixed-point implementations [12], since the computation is fixed-point with respect to each individual computational unit. However, by potentially allowing each signal in the original specification to be encoded by binary words with different scaling and wordlength, the degrees of freedom in design are significantly increased.

Definition 2: An *annotated computation graph* $G'(V, S, A)$ is a formal representation of a finite-wordlength implementation of computation graph $G(V, S)$. A is a pair $(\mathbf{n}, \mathbf{p})$ of vectors $\mathbf{n} \in \mathbb{N}^{|S|}$, $\mathbf{p} \in \mathbb{Z}^{|S|}$, each with elements in one-to-one correspondence with the elements of S. Thus, for each $s \in S$, it is possible to refer to the corresponding annotation (n_s, p_s). Note that throughout this paper we shall deal entirely with ℓ_1-scaled systems [2], i.e., those where the scaling vector $\mathbf{p}$ is predetermined from the system transfer functions to avoid overflow errors. Alternative scalings are considered elsewhere [13].

D. Wordlength Propagation and Conditioning

In order to predict the quantization effect of a particular wordlength and scaling annotation, it is necessary to propagate the wordlength values and scalings from the inputs of each atomic operation to the operation output, as shown in Table I. The "q" superscript is used to indicate a wordlength before quantization, i.e., truncation or rounding.

The wordlength values derived through format propagation may then be adjusted according to the known ℓ_1 scaling of the output signal. If the ℓ_1-scaled binary point location at signal s is p_s, whereas the propagated value derived is p'_s $(> p_s)$, then this corresponds to a most significant bit (MSB)-side-width reduction. An adjustment $n_s^q \leftarrow n_s^{q\prime} - (p'_s - p_s)$ must then be made, where $n_s^{q\prime}$ is the propagated wordlength value, as illustrated in Fig. 2. Conceptually, this operation is inverse sign-

TABLE I
PROPAGATION OF WORDLENGTHS

TYPE(v)	Propagation rules for $o \in \text{out}(v)$
GAIN	For input (n_i, p_i) and coefficient (n_j, p_j): $p'_o = p_i + p_j$ $n_o^{q\prime} = n_i + n_j$
ADD	For inputs (n_i, p_i) and (n_j, p_j): $p'_o = \max(p_i, p_j) + 1$ $n_o^{q\prime} = \max(n_i, n_j + p_i - p_j) - \min(0, p_i - p_j) + 1$ (for $n_i > p_i - p_j$ or $n_j > p_j - p_i$)
DELAY or FORK	For input (n_i, p_i): $p'_o = p_i$ $n_o^{q\prime} = n_i$

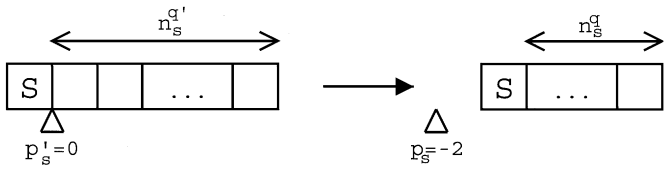

Fig. 2. Wordlength and scaling adjustment.

extension. This analysis allows correlation information derived from ℓ_1 scaling to effectively take advantage of a type of "don't-care condition" not usually considered by synthesis tools.

When designing a multiple wordlength implementation $G'(V, S, A)$, certain choices of A are clearly suboptimal. Consider, for example, a GAIN node which multiplies signal s of wordlength n_s by a coefficient of format (n, p). If the output signal s' has been assigned wordlength $n_{s'} > n_s + n$, then this assignment is suboptimal, since at most $n_s + n$ bits are necessary to represent the result to full precision. Ensuring that such cases do not arise is referred to as "conditioning" the annotated computation graph. Conditioning is an important design step, as it allows the search space of efficient implementations to be pruned, and ensures that the most efficient use is made of all bits of each signal. It is, thus, possible to define a *well-conditioned* annotated computation graph to be one in which there are no superfluous bits representing any signal.

Definition 3: An annotated computation graph $G'(V, S, A)$ with $A = (\mathbf{n}, \mathbf{p})$ is said to be *well-conditioned* if and only if $n_s \leq n_s^q$ for all $s \in S$.

During heuristic optimization, to be described in Section V, ill-conditioned annotated computation graphs may result as intermediate structures. An ill-conditioned annotated computation graph can always be transformed into an equivalent well-conditioned form in the iterative manner shown in Algorithm 1 . For a well-connected graph [3], this algorithm will always terminate.

```
Algorithm 1: Algorithm WlCondition
Input: An Annotated Computation Graph
 G'(V,S,A)
Output: An annotated computation graph,
 with well-conditioned wordlengths and
 identical behavior to the input system
begin
 Calculate p'_s and n_s^q' for all signals
 s ∈ S   (Table I)
 Form n_s^q from n_s^q', p'_s and p_s for all signals
 s ∈ S
 while ∃s ∈ S : n_s^q < n_s
   Set n_s ← n_s^q
   Update n_s^q' for all affected signals
   (Table I)
   Re-form n_s^q from n_s^q', p'_s and p_s for all
   affected signals
 end while
end
```

E. Noise Model

This section describes the noise model used within the wordlength optimization procedures. The noise model is novel in two respects. First, it does not make the usual assumption that the wordlength before quantization has a negligible effect on the statistical properties of the noise injected due to truncation or roundoff. (This is usually the case in a uniprocessor implementation due to the wordlength before quantization being much larger than the wordlength after quantization). Second, it treats FORK nodes in a special way, to ensure statistically uncorrelated error when different outputs of a FORK have different wordlength. Taken together, these novel aspects allow precise modeling of the complexities of the design space, including its nonmonotonicity and, indeed, nonconvexity [3].

At each point within the computation where truncation or rounding is performed, it is possible to estimate the variance of the injected noise analytically, using a discrete noise model introduced in [14]. The variance of such an error signal, when truncating from n_1 bits to n_2 bits with a scaling p, is given by

$$\sigma^2 = \frac{1}{12} 2^{2p} (2^{-2n_1} - 2^{-2n_2}). \tag{1}$$

For each signal $s \in \{(v_1, v_2) \in S : v_1 \notin V_F\}$ a straightforward application of (1) may be used with $n_1 = n_s^q$, $n_2 = n_s$, and $p = p_s$, where n_s^q is as defined in Section III-D.

Signals emanating from nodes of fork type must be considered somewhat differently. Fig. 3(a) shows one such annotated fork structure, together with possible noise models in Fig. 3(b) and (c). Either model is valid, however, Fig. 3(c) has the advantage that the error signals $e_0[t]$, $e_1[t]$, and $e_2[t]$ show very little correlation in practice compared to the structure of Fig. 3(b). This is due to the overlap in the low-order bits truncated in Fig. 3(b). Note also that the z-domain transfer functions from the noise inputs to the system outputs are different under the two models—this will be discussed in further detail in Section IV-C, where modeling of FORK-noise is considered.

As a result of the correlation, the *cascaded* model is preferred, as it allows L_2 scaling [2] to predict the propagation of error to the system outputs.

F. Component Libraries and Area Models

It is assumed, when constructing a cost model, that both a dedicated resource binding is to be used [15] and that the area cost of wiring is negligible, i.e., the designs are *resource dominated* [15]. These assumptions simplify the construction of an area-cost model, since it becomes sufficient to estimate separately the area consumed by each computation node, and then to sum the resulting estimates.

The problem of area modeling has been approached from a "black box" perspective, as use is made of highly optimized integer arithmetic cores available from FPGA manufacturers as the computational element at the center of each multiple wordlength component.

The area model for a multiple wordlength adder is reasonably straightforward. The ripple-carry architecture is used since FPGAs provide good support for fast ripple-carry implementations [16], [17]. Consider the multiple wordlength adders illustrated in Fig. 4. The adders have width $\max(n_a - f_v, n_b) - m_v + 2$ bits. Each bit may

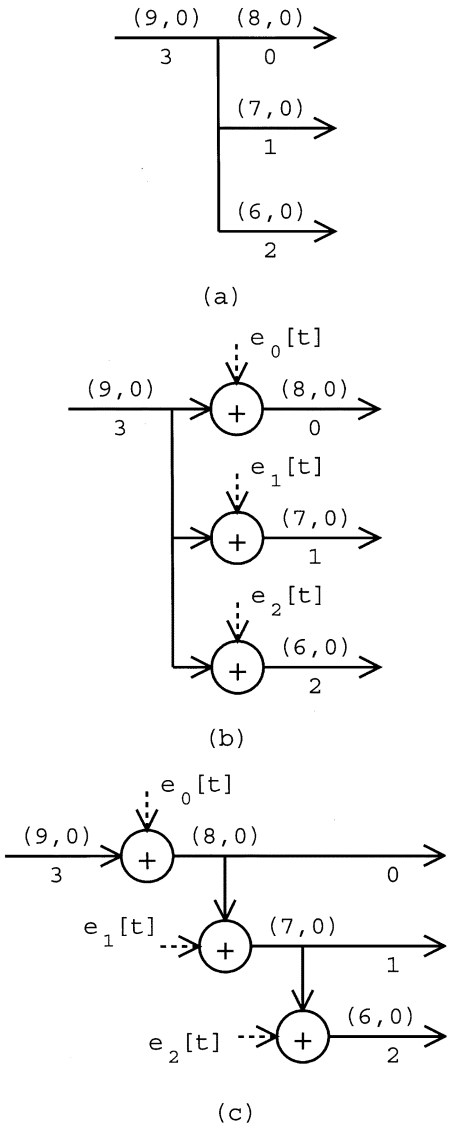

Fig. 3. Modeling post-FORK truncation.

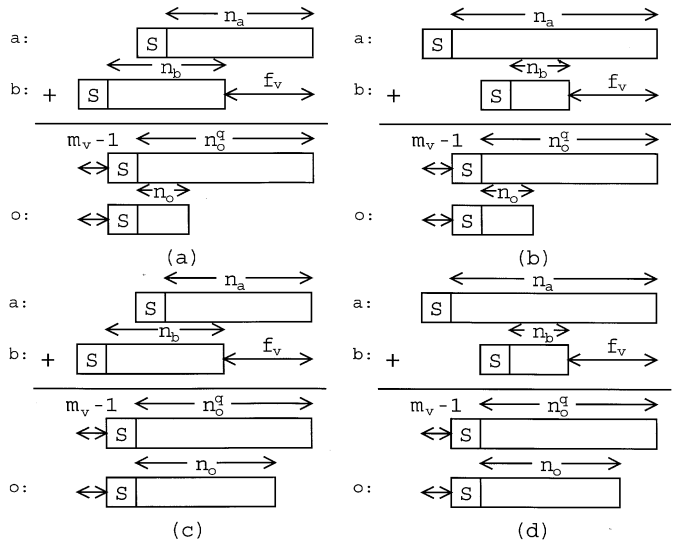

Fig. 4. Multiple wordlength adder formats. Each adder $v \in V_A$ has input signals a and b and output signal $o \in S$. The number of bits by which input b must be shifted left is denoted f_v, and the number of resulting MSBs which may be ignored due to ℓ_1-scaling is denoted m_v.

not consume the same area, however, because in Fig. 4(a) and (b) some bits are only required for carry propagation; their sum outputs are unused. The cost model, therefore, has two parameters k_1 and k_2 corresponding to the area cost of a sum-and-carry full adder, and the area cost of a carry-only full adder. Also, in Fig. 4(c) and (d), some of the result is drawn from the "overhang" of input a and may thus be obtained at no cost. Combining these observations, the area of an adder is expressed in (2), shown at the bottom of the page.

Area estimation for constant coefficient multipliers is more problematic, as a constant coefficient multiplier is typically implemented through recoding as a series of additions. This implementation style causes the area consumption to be dependent on coefficient value. In addition, modeling the exact implementation scheme used would make the model highly vendor-specific.

Instead, a simple area model has been constructed which models the linear area growth in both coefficient and data wordlength as well as computational elements required only for carry propagation. This is shown in (3) for a node $v \in V_G$ with input $i \in S$ and output $j \in S$. Recall that cw denotes coefficient wordlength. The coefficient values k_3 and k_4 have been determined through least-squared fitting to synthesis results from several hundred multipliers of different coefficient value and width.

$$A_v = k_3 cw(v)(n_i + 1) + k_4\Big(n_i + cw(v) - n_o\Big). \tag{3}$$

The area of a DELAY node $v \in V_D$ is simple, as the area consumed varies linearly with the input wordlength n_i, giving $A_v = k_5(n_i + 1)$.

G. Optimization Formulation

Combining the area models presented in Section III-F into a single area measure on G gives a cost metric $A_G(\mathbf{n}, \mathbf{p})$. Combining the L_2-norm error-variance models into a vector $\mathbf{E}_G(\mathbf{n}, \mathbf{p})$ with one element per output, allows the wordlength-optimization problem to be formulated in Problem 1 .

Problem 1 (Wordlength Optimization): Given a computation graph $G(V, S)$, and a scaling vector $\mathbf{p}$, the wordlength optimization problem may be defined as to select $\mathbf{n}$ such that $A_G(\mathbf{n}, \mathbf{p})$ is minimized subject to (4), where $\mathcal{E}$ denotes the user specified bound on error variance at each system output.

$$\begin{aligned} &\mathbf{n} \in \mathbb{N}^{|S|} \\ &\mathbf{E}_G(\mathbf{n}, \mathbf{p}) \leq \mathcal{E}. \end{aligned} \tag{4}$$

It has been shown that this constraint space is nonconvex in nature [3], and that the optimization problem is NP-hard [4]. In the following sections, two approaches to the above problem are proposed: modeling the problem as a mixed integer linear program (Section IV) and a heuristic technique (Section V).

$$A_v = \begin{cases} k_1(n_o + 1) + k_2(\max(n_a - f_v, n_b) - m_v - n_o + 1), & \text{if } n_o + m_v \leq \max(n_a - f_v, n_b) + 1 \\ k_1(\max(n_a - f_v, n_b) - m_v + 2), & \text{otherwise} \end{cases} \tag{2}$$

IV. MILP Model

The proposed MILP model contains several variables, which may be classified as: integer signal wordlengths (n), integer signal wordlengths before quantization (n^q), binary auxiliary signal wordlengths ($\bar{n}$), binary auxiliary signal wordlengths before quantization ($\bar{n}^q$), binary decision variables (δ, ϵ, η), real adder costs (A), integer adder auxiliary variables (β), and real fork node errors (E). Each of these variable types and their uses will be described below.

Note that only ADD, GAIN, and DELAY nodes consume area resources (FORK nodes are considered free). However, adders have an inherently nonlinear area model and thus while gains and delays are included directly in the objective function, the cost of each adder $v \in V_A$ is represented by a distinct variable A_v, introduced in Section IV-B.

An area-based objective function for the MILP model may then be formulated (5), as described in Section III-F: the three terms present represent the adder area, gain area, and delay area, respectively. Constant terms from the GAIN and DELAY area models are not included in the objective function, as they play no role in the optimization.

$$\min: \quad \sum_{v\in V_A} A_v + \sum_{v\in V_G}\left\{\left(k_3 cw(v)+k_4\right)n_{\mathrm{in}(v)} + k_4 n_{\mathrm{out}(v)}\right\} + \sum_{v\in V_D} k_5 n_{\mathrm{in}(v)} \quad (5)$$

Constraints on quantization error propagation are much harder to cast in linear form due to the exponentiation in the noise model (Section III-E). In order to overcome this nonlinearity, we propose to use auxiliary binary variables, $\bar{n}_{s,b}$, one for each possible wordlength b that a signal s could take. By using these binary variables, it is possible to recast expressions of the form 2^{-2n_s}, which appear in error constraints, into linear form as $\sum_{b=1}^{\hat{n}_s} 2^{-2b}\bar{n}_{s,b}$.

The relationship between the auxiliary variables and the wordlength variables that is expressed in (6) and (7) ensures that each signal can only have a single wordlength value. Note that in order to apply this technique, it is necessary to know the upper-bound wordlengths $\hat{n}_s$ for each $s \in S$. Derivation of these bounds will be discussed in Section IV-A. Note also that signals driving fork nodes are not considered in this way; fork node error models are considered separately due to the cascade model, as described in Section IV-C

$$\forall s\in S\backslash \mathrm{in}(V_F),\ n_s - \sum_{b=1}^{\hat{n}_s} b\cdot\bar{n}_{s,b} = 0 \quad (6)$$

$$\forall s\in S\backslash \mathrm{in}(V_F), \qquad \sum_{b=1}^{\hat{n}_s}\bar{n}_{s,b} = 1. \quad (7)$$

In a similar manner, it is necessary to linearize the exponentials in wordlengths before quantization (8) and (9). This time signals driven by fork nodes are not considered as, from the cascade model, the wordlength before quantization of any one of these signals is simply the wordlength of the preceding signal in the cascade. Similarly, signals driven by INPORT nodes are not considered, as their prequantization wordlength is defined completely by the system environment.

$$\forall s \in S\backslash \mathrm{in}(V_F)\backslash \mathrm{out}(V_F)\backslash \mathrm{out}(V_I),\ n_s^q - \sum_{b=1}^{\hat{n}_s^q} b\bar{n}_{s,b}^q = 0 \quad (8)$$

$$\forall s \in S\backslash \mathrm{in}(V_F)\backslash \mathrm{out}(V_F)\backslash \mathrm{out}(V_I), \sum_{b=1}^{\hat{n}_s^q}\bar{n}_{s,b}^q = 1. \quad (9)$$

For each system output, we propose to use an error constraint of the form given in (10). This inequality derives from a direct combination of the noise model described in Section III-E with the linearization process described above. Note that for simplicity of explanation, only single-output systems are considered in this section, however, the technique is easily generalizable to multiple-output (MIMO) systems.

Note that those signals driven by system inputs are considered separately [third term of (10)], since there is no need for Boolean variables representing the prequantization wordlength of a variable, as this parameter is defined by the system environment. Placeholders E_v are used for the contribution from fork nodes [first term of (10)]. These will be defined by separate constraints in Section IV-C due to the cascade model.

$$\sum_{v\in V_F} E_v + \sum_{s\in S\backslash \mathrm{in}(V_F)\backslash \mathrm{out}(V_F)\backslash \mathrm{out}(V_I)} 2^{2p_s} L_2^2\{H_s(z)\} \times \left(\sum_{b=1}^{\hat{n}_s} 2^{-2b}\bar{n}_{s,b} - \sum_{b=1}^{\hat{n}_s^q} 2^{-2b}\bar{n}_{s,b}^q\right) + \sum_{s\in \mathrm{out}(V_I)} 2^{2p_s} L_2^2\{H_s(z)\}\left(\sum_{b=1}^{\hat{n}_s} 2^{-2b}\bar{n}_{s,b} - 2^{-2n_s^q}\right) \le 12\mathcal{E}. \quad (10)$$

A. Wordlength Bounds

Upper bounds on the wordlength of each signal, before and after quantization, are required by the MILP model in order to have a bounded number of binary variables corresponding to the possible wordlengths of a signal.

The proposed bounding procedure consists of three stages: performing a heuristic wordlength optimization on the computation graph using the heuristic to be described in this paper; using the resulting area as an upper bound on the area of each gain block within the system, and hence, on the input wordlength of each gain block; and "conditioning" the graph, following Algorithm 1 (Section III-D). The intuition is that the bulk of the area consumed in a DSP implementation typically comes from multipliers. Thus, reasonable upper bounds are achievable by ensuring that the cost of each single multiplier cannot be greater than the heuristically achieved cost for the entire implementation. Of course, this only bounds the wordlength of signals which drive gain blocks. In addition, the wordlength of signals driven by primary inputs is bounded by the externally defined precision of these inputs. Together, this information can be propagated through the computation graph by Algorithm 1, resulting in upper bounds for all signals under the condition that any closed loop must contain a gain block.

We denote by $\hat{n}_s$ the so-derived upper bound on the wordlength of signal $s \in S$ and by $\hat{n}_s^q$ the upper bound on the wordlength of the same signal before LSB truncation.

B. Adders

It is necessary to express the area model of Section III-F as a set of constraints in the MILP. Also, a set of constraints describing how the wordlength at an adder output varies with the input wordlengths is required.

1) Area Model: In the objective function (5), the area for each adder $v \in V_A$ was modeled by a single variable A_v. It will be demonstrated in this section how this area can be expressed in linear form.

Let us define β_v for an adder $v \in V_A$ with input signals a and b by (11), where the inputs "a" and "b" are chosen to match with Fig. 4 so that it is b which needs to be left-shifted for alignment purposes. Recall that f_v is also illustrated in Fig. 4, and models the number of bits by which input b must be shifted

$$\beta_v = \max(n_a - f_v, n_b). \quad (11)$$

This definition is used to simplify the expression for the area of an adder, from Section III-F, as

$$A_v = \begin{cases} k_1(n_o+1)+k_2[\beta_v - m_v - n_o + 1], & \text{if } n_o + m_v \le \beta_v + 1 \\ k_1[\beta_v - m_v + 2], & \text{otherwise} \end{cases}. \tag{12}$$

The value of m_v is independent of the wordlengths, and for an adder can be expressed as $m_v = \max(p_a, p_b) + 1 - p_o$.

The nonlinearities due to the max operator in (11), the decision in (12), and the choice of which input needs to be left-shifted for alignment, must all be linearized for the MILP model. This is achieved through the introduction of four binary decision variables $\delta_{v1}, \delta_{v2}, \delta_{v3}$, and δ_{v4} for each adder $v \in V_A$, to be described in the following paragraphs.

For the remainder of this section, we consider a general adder with inputs i and j and output o, to distinguish from the more specific case considered above, where input b was used to denote the left-shifted input to an adder.

In order to linearize (11), if $p_j \le p_i$, then it is sufficient to include (13)–(16) in the MILP. Otherwise, (17)–(20) should be included in the MILP. These two cases derive from differing values of m_v, which can be determined statically during MILP construction rather than MILP solution. The right-hand side of each inequality (13)–(20) consists of a trivial bound on the left hand side, multiplied by a binary decision variable. These constraints are used in order to allow disjunctions and, thus, implications, for example selecting $\delta_{v1} = 0$ in (13) gives $n_i - n_j + p_j - p_i < 0$, whereas selecting $\delta_{v1} = 1$ gives $\beta_v - n_j + p_j - p_i \ge 0$. Allowing choice of δ_{v1} as an optimization variable results in the implication $n_i \ge n_j - p_j + p_i \Rightarrow \beta_v \ge n_j + p_j - p_i$, as the other constraints are trivially satisfied. [This construction corresponds to one of the two cases of the "max" operator in (11)].

$$n_i - n_j + p_j - p_i < \delta_{v1}(\hat{n}_i + p_j - p_i) \tag{13}$$
$$\beta_v - n_j + p_j - p_i \ge (1-\delta_{v1})(-\hat{n}_j - p_i + p_j) \tag{14}$$
$$n_i - n_j + p_j - p_i \ge \delta_{v2}(-\hat{n}_j + p_j - p_i) \tag{15}$$
$$\beta_v - n_i \ge (1-\delta_{v2})(-\hat{n}_i) \tag{16}$$
$$n_j - n_i + p_i - p_j < \delta_{v1}(\hat{n}_j - p_j + p_i) \tag{17}$$
$$\beta_v - n_i + p_i - p_j \ge (1-\delta_{v1})(1 - \hat{n}_i - p_j + p_i) \tag{18}$$
$$n_j - n_i + p_i - p_j \ge \delta_{v2}(-\hat{n}_i + p_i - p_j) \tag{19}$$
$$\beta_v - n_j \ge (1-\delta_{v2})(-\hat{n}_j). \tag{20}$$

Note that β_v is only bounded from below by the constraint given. The equality in (11) is guaranteed through the positive coefficient of A_v in the objective function.

In order to linearize (12), inequalities (21)–(24) are included in the MILP. These constraints model the choice in (12) as a pair of implications, in an identical manner to that described above.

$$n_o - \beta_v + m_v - 1 \ge \delta_{v3}(m_v - \hat{\beta}_v) \tag{21}$$
$$A_v + (k_2 - k_1)n_o - k_2\beta_v + k_2(m_v - 1) - k_1 \ge (1-\delta_{v3})\left[(k_2-k_1)\hat{n}_o - k_2\hat{\beta}_v + k_2(m_v-1) - k_1\right] \tag{22}$$
$$n_o - \beta_v + m_v - 1 < \delta_{v4}(\hat{n}_o + m_v - 2) \tag{23}$$
$$A_v + k_1(m_v - \beta_v - 2) \ge (1-\delta_{v4})k_1(m_v - \hat{\beta}_v - 2). \tag{24}$$

2) Output Wordlength: The prequantization output wordlength of an adder with inputs i and j and output o is given by $n_o^q = \max(n_i - p_i, n_j - p_j) + p_o$. We may express this as (25) and (26), since prequantization wordlengths only appear with negative coefficient in the error terms, so the error constraints can be relied upon to reduce n_o^q to achieve equality

$$n_o^q \ge n_i - p_i + p_o \tag{25}$$
$$n_o^q \ge n_j - p_j + p_o. \tag{26}$$

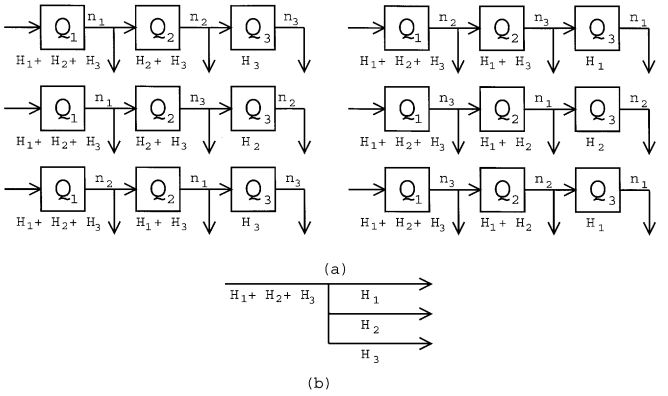

Fig. 5. Output permutations in a three-way fork.

C. Forks

In systems containing a reconverging fork node, complex error behavior can occur due to the different possible orderings of wordlength at the fork output, under the cascade error model introduced in Section III-E. Fig. 5(a) illustrates the six different possible configurations of a three-way fork with outputs n_1, n_2, and n_3. For example, the top-left figure corresponds to $n_1 \ge n_2 \ge n_3$ and the bottom-right figure to $n_3 \ge n_2 \ge n_1$. Each of the "Q" blocks is a truncation of least-significant bits in a signal. If we denote by H_i the z-domain transfer function from signal i to the output in the original specification [Fig. 5(b)], then the z-domain transfer function from the truncation error injected to the system output is shown underneath the relevant "Q" block in Fig. 5(a) [2].

In order for the MILP to fully model this behavior, it is necessary to consider each of the possible orderings. Let σ_v be a w-tuple, representing an order on a w-way fork node $v \in V_F$ with input signal $i \in S$. Thus, for example, $\sigma_v(2)$ is the second largest signal width. We may express the error resulting from truncation of those signals driven by node v as (27), with one constraint per possible ordering, a total of $w!$. Here, $\wedge$ represents Boolean conjunction, and $H_s(z)$ is the z-domain transfer function from signal $s \in S$ to the output in question.

$$\bigwedge_{r=1}^{w-1}(n_{\sigma_v(r)} \ge n_{\sigma_v(r+1)}) \Rightarrow E_v = 2^{2p_i}\left(\sum_{r=1}^{w-1} L_2^2\left\{\sum_{h=1}^{w-r} H_{\sigma_v(h)}\right\} \times (2^{-2n_{\sigma_v(r+1)}} - 2^{-2n_{\sigma_v(r)}}) + L_2^2\left\{\sum_{h=1}^{w} H_{\sigma_v(h)}(2^{-2n_i^q} - 2^{-2n_w})\right\}\right) \tag{27}$$

Applying DeMorgan's theorem and linearizing the resulting disjunction gives (28)–(32). Each exponential is then further linearized using the binary auxiliary variables. The ϵ and η variables in (28)–(32) are additional binary decision variables and the right-hand side of each inequality consists of a trivial bound on the left-hand side, multiplied by

a decision variable. At least one inequality is nontrivial, a property ensured by (32)

$$n_{\sigma(1)} - n_{\sigma(2)} < \epsilon_{v\sigma(1),\sigma(2)}\hat{n}_{\sigma(1)} \tag{28}$$

$$n_{\sigma(2)} - n_{\sigma(3)} < \epsilon_{v\sigma(2),\sigma(3)}\hat{n}_{\sigma(2)} \tag{29}$$

$$\cdots$$

$$n_{\sigma(w-1)} - n_{\sigma(w)} < \epsilon_{v\sigma(w-1),w}\hat{n}_{\sigma(w-1)} \tag{30}$$

$$\begin{aligned} E_v - 2^{2p_i} &\times \left(\sum_{r=1}^{w-1} L_2^2 \left\{ \sum_{h=1}^{w-r} H_{\sigma(h)} \right\} \right. \\ &\times (2^{-2n_{\sigma_v(r+1)}} - 2^{-2n_{\sigma_v(r)}}) \\ &\left. + L_2^2 \left\{ \sum_{h=1}^{w} H_{\sigma(h)} \right\} (2^{-2n_{\sigma_v(w)}} - 2^{-2n_o^q}) \right) \\ &\geq -\eta_{v\sigma} 2^{2(p_i-1)} \sum_{r=0}^{w-1} L_2^2 \left\{ \sum_{h=1}^{w-r} H_{\sigma(h)} \right\} \end{aligned} \tag{31}$$

$$\sum_{r=1}^{w-1} \epsilon_{v\sigma(r),\sigma(r+1)} + \eta_{v\sigma} \leq w - 1. \tag{32}$$

It is not necessary to explicitly consider quantization of the input signal to a fork node, since the above constraints use the prequantization wordlength of the fork input n_i^q. It is necessary, however, to guarantee that the input signal provides enough wordlength for the largest of its outputs (33)

$$n_i \geq n_a, n_i \geq n_b, \ldots, n_i \geq n_f. \tag{33}$$

D. Gains

In contrast to adders and fork nodes, modeling of gain nodes is straightforward. The area of a gain node is linear and, thus, has already been modeled in the objective function (Section IV). The only remaining constraint required is to model the prequantization output wordlength of a gain $v \in V_G$ with input signal a, output signal o, coefficient of wordlength $cw(v)$, and scaling $sc(v)$. This constraint is naturally in linear form (34), and thus needs no further attention

$$n_o^q = n_a + cw(v) - p_a - sc(v) + p_o. \tag{34}$$

E. Delays

Delay nodes also have a simple relationship between their input wordlength and their output wordlength before quantization, shown in (35) for the case of a delay node with input i and output o.

$$n_o^q = n_i. \tag{35}$$

F. MILP Summary

An MILP model for the wordlength optimization problem has been proposed. This section collects the results from previous paragraphs, in order to quantify the number of variables (36) and constraints (37) present in the model. Each term in these equations is commented with the variables or constraints to which it refers (a semicolon denotes the comment). Note that the number of constraints given does not include integrality constraints, the unit upper bounds on Boolean variables, or slack variables introduced by the MILP solver

$$\begin{aligned} \text{vars} = & \sum_{s \in S \setminus \text{in}(V_F)} (\hat{n}_s + 1) +; (\bar{n}_{s,b} \text{ and } n_s) \\ & \sum_{s \in S \setminus \text{in}(V_F) \setminus \text{out}(V_F) \setminus \text{out}(V_I)} (\hat{n}_s^q + 1) +; (\bar{n}_{s,b}^q \text{ and } n_s^q) \\ & |V_F| +; (E_v) \\ & 6|V_A| +; (\delta_{v1}, \ldots, \delta_{v4}, \beta_v, A_v) \\ & \sum_{v \in V_F} od(v)(od(v)-1)\{1 + (od(v)-2)!\} \\ & ; (\epsilon \text{ and } \eta) \end{aligned} \tag{36}$$

$$\begin{aligned} \text{cons} = & 2|S \setminus \text{in}(V_F)| +; (6) \text{ and } (7) \\ & 2|S \setminus \text{in}(V_F) \setminus \text{out}(V_F) \setminus \text{out}(V_I)| +; (8) \text{ and } (9) \\ & 1 +; (10) \\ & 10|V_A| +; [(13)\text{–}(16) \text{ or } (17)\text{–}(20)] \text{and } (21)\text{–}(24) \\ & \sum_{v \in V_F} od^2(v) + 2od(v)(od(v)-1)! +; (28)\text{–}(33) \\ & |V_G| + |V_D|; (34) \text{ and } (35). \end{aligned} \tag{37}$$

It can be seen that so long as the number of large-fanout fork nodes is limited, the number of constraints in the MILP model grows approximately linearly in the number of nodes and signals. Under the same conditions the number of variables can grow up to quadratically with the number of signals because the upper bounds on each signal wordlength will vary approximately linearly with the number of large area-consuming nodes. Both parameters are dominated by any large-fanout fork nodes, since the number of η variables and their associated constraints grow combinatorially in the fanout.

Results from the MILP approach will be presented in Section VI.

V. Heuristic Approach

The proposed heuristic is shown in Algorithm 2. After performing an ℓ_1 scaling, the algorithm determines the minimum uniform wordlength satisfying all error constraints. The design at this stage corresponds to a standard uniform wordlength design with implicit power-of-two scaling, such as may be used for an optimized uniprocessor implementation. Each wordlength is then scaled up by a factor $k > 1$, which represents a bound on the largest value that any wordlength in the final design may reach. In the Synoptix implementation, $k = 2$ has been used. At this point, the structure may be ill-conditioned, requiring reduction to a well-conditioned structure, as described in Section III-D.

The resulting well-conditioned structure forms a starting point from which one signal wordlength is reduced by one bit on each iteration. The signal wordlength to reduce is decided in each iteration by reducing each wordlength in turn until it violates an output noise constraint. At this point, there is likely to have been some payoff in the reduced area, and the signal whose wordlength reduction provided the largest payoff is chosen. The wordlength of each signal is explored using a binary search.

Although Algorithm 2 is a greedy algorithm, both the constraints and the objective function play a role in determining the direction of movement toward the solution. As a result, this algorithm is less dependent on local information than a pure steepest-descent search, such as the "heuristic procedure" of [11]. Experiments show this to result in improvements of 5%–6% in area over a pure steepest descent approach.

Algorithm 2 will, in general, provide better results under a convex constraint space. However, the intuition is that any nonconvexity present in the space should not affect its operation too severely: the binary search mechanism will result in "jumping over" infeasible portions of constraint space in some cases, and will remain stuck on

one side of an infeasible region in other cases. In either case, since the wordlengths are only reduced by one bit each time, the next iteration is likely to face the nonconvexity from a different direction. Nonconvexity can manifest itself in intermediate wordlength vectors having infeasible error properties, however, the final wordlength vector will always be feasible as moves are only ever performed in a direction leading to a feasible solution, and the cost metric $A_G(\mathbf{n}, \mathbf{p})$ is monotonic in $\mathbf{n}$.

```
Algorithm 2: Algorithm WLOptHeur
Input: A Computation Graph G(V,S)
Output: an optimized annotated computation
  graph G'(V,S,A), with A = (n,p)
begin
 Let the elements of S be denoted as
 S = {s_1, s_2, ... s_|S|}
 Determine p through l_1 scaling
 Determine u, the minimum uniform
 wordlength satisfying (4) with n = u·1
 Set n ← ku·1
 do
  Condition the graph G'(V,S,A)
  (Algorithm 1)
  Set Currentcost ← A_G(n,p)
  foreach signal s_i ∈ S do
   Set bestmin ← currentcost
   Determine w ∈ {2,...,n_si}, if such a w
   exists, such that (4) is satisfied for
   annotation
   ([n_s1 ... n_si-1 w n_si+1 ... n_s|S|], p) but not
   satisfied for annotation ([n_s1 ... n_si-1
   (w − 1) n_si+1 ... n_s|S|], p)
   If such a w exists, set
    minval ← A_G([n_s1 ... n_si-1 w n_si+1 ... n_s|S|], p)
   If no such w exists, set
    minval ← A_G([n_s1 ... n_si-1 1 n_si+1 ... n_s|S|], p)
   if minval < bestmin do
    Set bestsig ← s_i, bestmin ← minval
   end if
  end foreach
  if bestmin < currentcost
    n_bestsig ← n_bestsig − 1
 while bestmin < currentcost
  end if
end
```

VI. Results

Synoptix, a complete synthesis system incorporating the algorithms in this paper, has been developed for implementation of multiple wordlength systems in hardware. The input to Synoptix is a Simulink [18] block diagram, and the output is a structural description in VHDL. Third-party tools are then used to perform the low-level logic synthesis, placement, and routing of the designs. The design-flow for implementation on the Sonic platform [19] is illustrated in Fig. 6, with the Sonic-specific parts shaded.

The system has been tested on several benchmark circuits, including FIR and IIR filters, a discrete cosine transform (DCT), a polyphase filter bank (PFB), and an RGB to YCrCb converter.

A. MILP Results

Fig. 7 illustrates area-error tradeoff curves for both a second- and a third-order linear phase FIR filter [2]. For the second-order filter, results for both four-bit and eight-bit inputs are given. For the third-order filter, only results for a four-bit input have been obtained. Three curves are present in each plot: the optimum uniform wordlength implementation, the heuristically derived multiple wordlength implementation from Section V, and the optimum multiple wordlength implementation achieved by solving the MILP presented in Section IV.

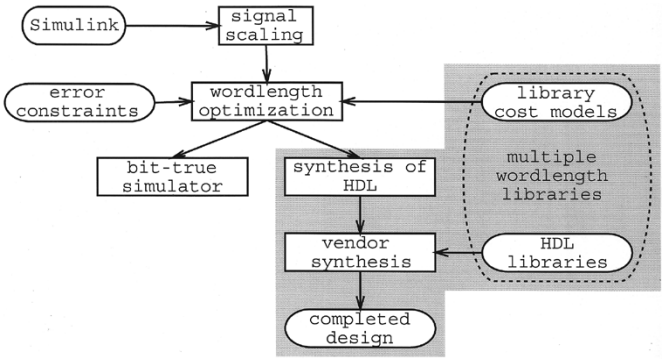

Fig. 6. Synoptix design flow.

The results clearly illustrate the high-quality solutions achievable by the heuristic solution, averaging only 0.7% with a maximum of 3.9% worse than the optimum result. Thus while finding optimum solutions for the general case is NP-hard, for practical examples the heuristic can achieve near-optimal results (with respect to the area and error estimates).

As a concrete example, an optimum wordlength allocation for an RGB to YCrCb converter described in [1] with four-bit inputs has also been performed. This result shows an optimal cost of 79 logic cells (four-input lookup tables), equal to the result achieved by the heuristic. Fig. 8 illustrates the structure [20] and wordlengths of the RGB to YCrCb converter for four-bit inputs (of range ± 112), four-bit coefficients, and with an error-free Y, whereas a bounded error variance of up to 10^{-2} has been allowed for Cr and Cb. We believe such results, even for small circuits, to be valuable as a benchmark against which any new approaches could be compared.

The BonsaiG MILP solver [21] was used to solve the MILP models: execution time ranged from 2 s to 6 min on a 512-MB RAM 1.2-GHz AMD Athlon. This compares to less than 0.2 s for the heuristic solutions on the same machine. Limits on the scale of the MILP solvable are due to both excessive runtime and numerical instabilities in the MILP solver.

B. Heuristic Results

Shown in Fig. 9 is a graph of area (measured in Altera logic cells [16]) against specified error variance. This plot is representative in terms of the general shape of the plots obtained for all designs. The benchmark is a simple second-order (biquadratic) IIR digital filter. Both the multiple wordlength design and the optimized uniform wordlength structure are shown. The plot of area for a uniform wordlength decreases in steep steps. This is because there is a sudden change when the next lowest wordlength becomes feasible with respect to the error constraints. This is not the case for the optimized multiple wordlength structures, since there are many more optimization variables and, hence, many different error powers are achievable. In addition, the heuristic line lies consistently below the uniform line (by 2% to 15%), showing a consistent area saving for this design.

Table II illustrates some further results from larger benchmark circuits. Both the number of logic cells (LCs) and maximum clock frequency are reported. Each of these results corresponds to a single point on the area-error tradeoff curve for the circuit, and have been placed and

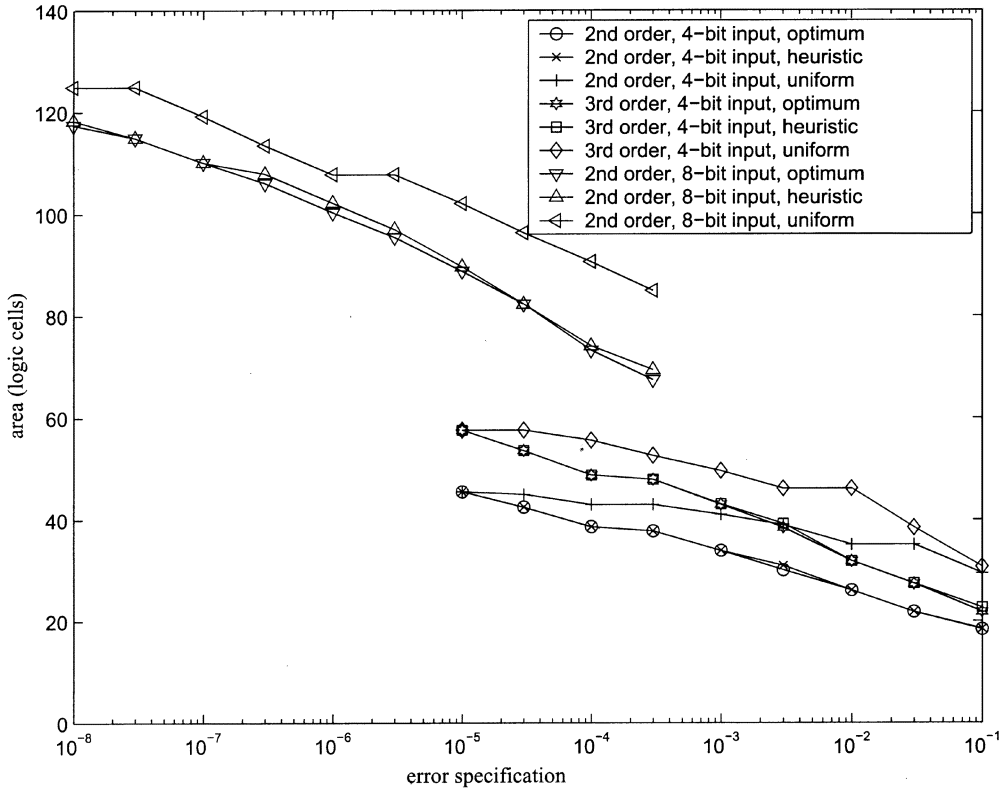

Fig. 7. Area/Error tradeoffs compared for a second- and third-order FIR filter.

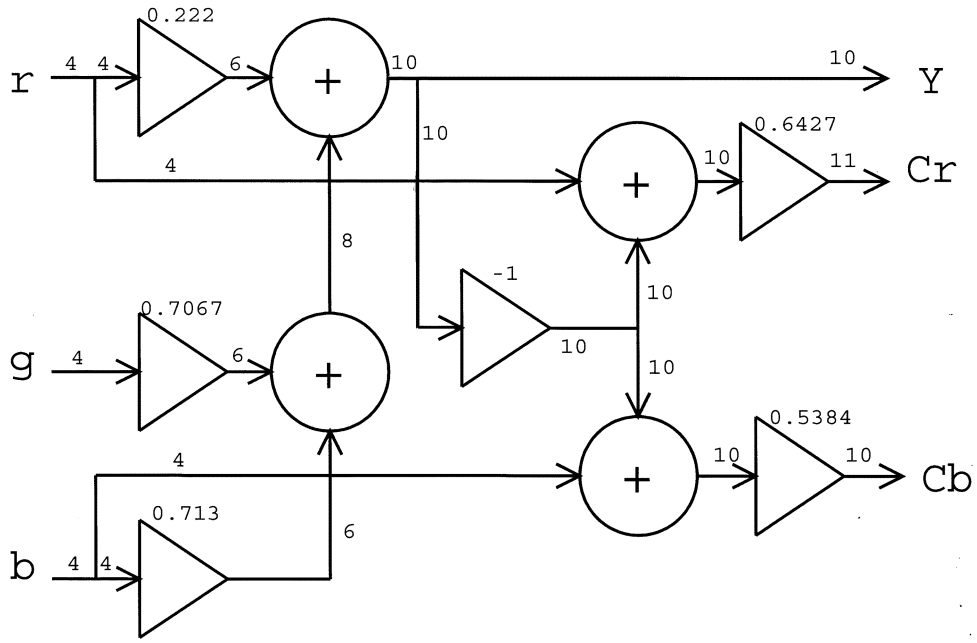

Fig. 8. Optimal wordlength allocations for the ITU RGB to YCrCb converter.

routed in an Altera Flex10k70RC240–3 device (as used in the Sonic [19] platform) except where otherwise stated. This device is representative of four-input lookup-table-based architectures, which form the core logic fabric of most FPGAs.

The FIR filter is a 126-tap linear-phase low-pass Direct Form II transposed [2] structure, suggested by [22] as a representative DSP design. The DCT is an eight-point, one-dimensional decimation-in-time structure from [23] which has also been suggested as a benchmark by [22]. Two versions of this benchmark have been synthesized, one (DCT^1) with equal error tolerance on all outputs, and the other (DCT^2) with required SNR reducing by 3 dB per DCT coefficient, so that low-frequency coefficients are less noisy than high-frequency ones. The IIR filter is of the fourth order, as used by [24] and is of interest since it has a recursive structure. The PFB is the design given in [25] for evaluation of the Streams-C compiler. The RGB to YCrCb converter is of the form suggested by the ITU [20], and allows some

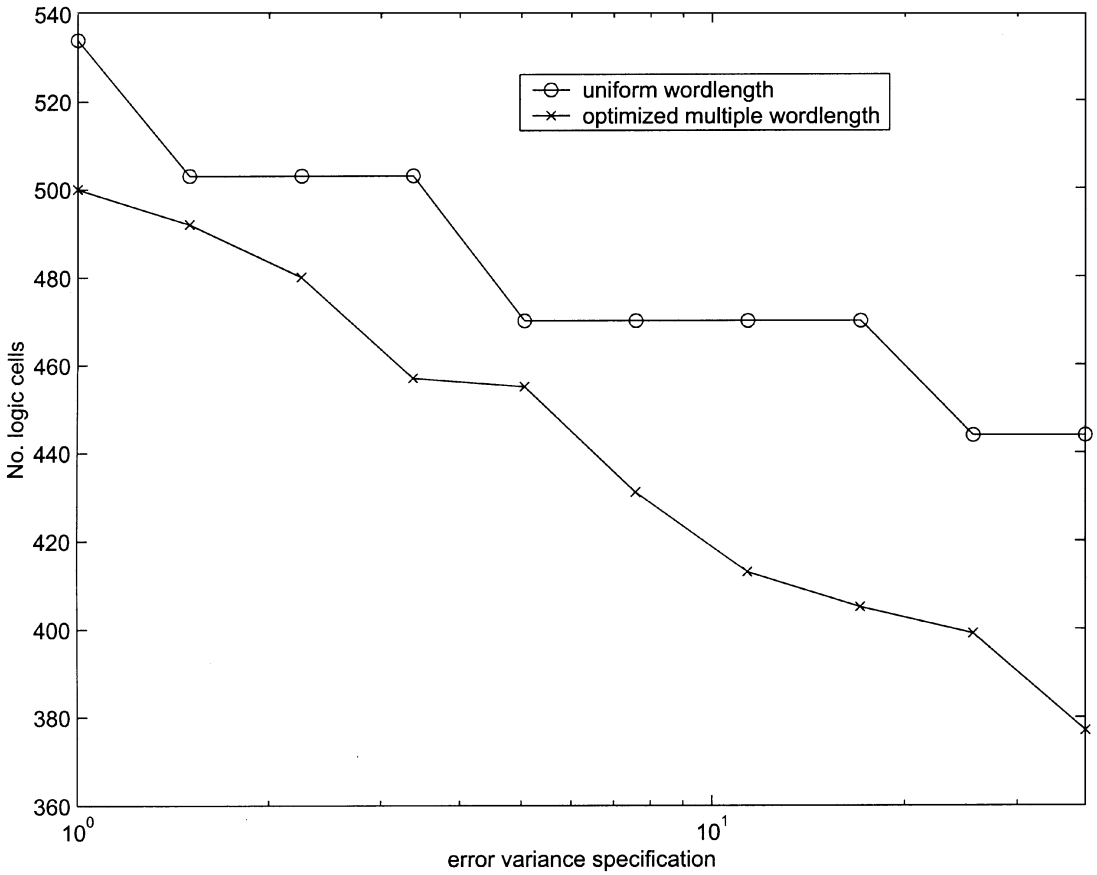

Fig. 9. Circuit area against specified error power for an IIR biquadratic filter.

TABLE II
LOSSY SYNTHESIS RESULTS

Design	Uniform wordlength			Multiple wordlength			
	Area (#LCs)	f_{clk} (MHz)	width (bits)	Area		f_{clk}	
				(#LCs)	% improvement	(MHz)	% improvement
FIR†	6125	29.15	16	3356	45.2%	36.23	2.5%
DCT1*	1394	12.95	13	1311	6.0%	13.67	5.6%
DCT2*	1367	13.03	12	1164	14.9%	13.53	3.8%
IIR	701	9.57	12	623	11.1%	9.32	-2.6%
PFB	321	30.03	15	273	15.0%	31.34	4.4%
RGB–YCrCb	438	11.58	18	272	37.9%	16.15	39.5%

* implemented on Flex10k70GC503-3
† implemented on Flex10k200SRC240-1

quantization error in the Cr and Cb outputs whereas the Y output is guaranteed to be error-free. This design is of particular interest since the multiple wordlength approach can clearly be used to customize the datapath in order to achieve these differential specifications. Each of these circuits has been synthesized twice, once using an optimal uniform wordlength structure, and once using the multiple wordlength structure generated by the Synoptix tool. The DCT designs have been synthesized on a device with a larger number of I/O pins, due to the I/O-limited nature of the designs, whereas the FIR filter has been synthesized on a device with a significantly larger logic capacity.

It should be noted that even for the uniform wordlength structures, Synoptix has been used to automatically insert power-of-two scaling [26], which is good practice in DSP design. Also, note that for both uniform and multiple wordlength structures, these circuits represent a completely unpipelined implementation of the specification, in order to aid direct comparison of maximum clock rate f_{clk} reported.

Table II illustrates that area reductions of between 6% and 45% (mean 22%) have been achieved by using the multiple-wordlength synthesis approach described in this paper. These area reductions have been accompanied by a speedup in maximum clock frequency between -3% and 39% (mean 12%), even though the estimated speed is not considered by the cost function used for optimization. Interestingly, the only benchmark to have been slowed down slightly as a result of the optimization is the IIR filter. This is due to the increase of some signal wordlengths on the critical path around the feedback loops in this filter. Importantly, the largest area reductions and speedups have occurred in both the FIR filter, which is the largest design shown, and the RGB to YCrCb converter, which has a structure ideally suited to multiple wordlengths since the error-free Y is calculated first, from which Cr and Cb are derived [20].

For all results, execution time ranged from 0.03 s to 15 min 57 s on a 512-MB RAM 1.2-GHz AMD Athlon.

VII. Conclusion

This paper has introduced a method for automating the design of bit-parallel multiple wordlength implementations of linear time-invariant DSP systems. A lossy synthesis approach has been described, based on optimizing the area consumption of the resulting implementation, subject to constraints on the finite precision errors. The techniques presented have been implemented in the Synoptix synthesis tool, which takes a Simulink block diagram as input and generates a structural hardware description language implementation.

It has been demonstrated that the multiple wordlength design paradigm allows a broad design space to be searched by the synthesis tools, leading to high quality results. An approach to obtain optimum solutions (with respect to the area and error models) based on mixed integer linear programming has been introduced, together with a heuristic alternative. Results have been presented from the application of the heuristic to benchmark DSP circuits, which show area reductions of up to 45% (mean 22%) and speed increases of up to 39% (mean 12%) compared with more traditional design techniques. The heuristic has been shown to come within 0.7% of the optimum area estimate for small benchmarks.

Although not demonstrated in this paper, it should also be noted that limit cycle behavior [2] is generally less problematic in multiple wordlength architectures than in their uniform wordlength equivalents [3].

The construction of the MILP is described in detail in this paper, however, no complete example MILP is given for space reasons. Several examples can be found at: http://infoeng.ee.ic.ac.uk/~gac1/OptimumWL.

Only LTI systems have been considered, restricting the domain of application, but also allowing the use of fast and accurate analytical error estimation, leading to a practical synthesis tool. We are currently investigating extensions of the described approach for some classes of nonlinear system [27]. In addition, we are incorporating resource-sharing models into our optimization framework. Further work could address the problem of wordlength optimization for different granularities of arithmetic component, such as those embedded within some recent FPGAs [17], as well as incorporating alternative arithmetic architectures, such as redundant arithmetic.

Acknowledgment

The authors wish to acknowledge helpful proposals from the anonymous reviewers.

References

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

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

[3] G. A. Constantinides, "High level synthesis and word length optimization of digital signal processing systems," Ph.D. dissertation, Elect. Electron. Eng., Univ. London, London, U.K., 2001.

[4] G. A. Constantinides and G. J. Woeginger, "The complexity of multiple wordlength assignment," *Appl. Math. Lett.*, vol. 15, no. 2, pp. 137–140, 2002.

[5] S. A. Wadekar and A. C. Parker, "Accuracy sensitive word-length selection for algorithm optimization," in *Proc. Int. Conf. Computer Design*, Austin, TX, Oct. 1998, pp. 54–61.

[6] 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, pp. 921–930, Aug. 2001.

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

[8] M.-A. Cantin, Y. Savaria, and P. Lavoie, "An automatic word length determination method," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2001, pp. V-53–V-56.

[9] M. Stephenson, J. Babb, and S. Amarasinghe, "Bitwidth analysis with application to silicon compilation," in *Proc. SIGPLAN Program. Lang. Design Implementation*, Vancouver, BC, Canada, June 2000, pp. 108–120.

[10] 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 Test Eur.*, Munich, Germany, 1999, pp. 271–276.

[11] M.-A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2002, pp. II-612–II-615.

[12] C. Inacio and D. Ombres, "The DSP decision: Fixed point or floating?," *IEEE Spectrum*, vol. 33, pp. 72–74, Sept. 1996.

[13] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Synthesis of saturation arithmetic architectures," *ACM Trans. Design Automation Electron. Syst.*, vol. 8, no. 3, 2003.

[14] ——, "Truncation noise in fixed-point SFG's," *IEE Electron. Lett.*, vol. 35, no. 23, pp. 2012–2014, 1999.

[15] G. DeMicheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw-Hill, 1994.

[16] *Altera Databook*, Altera Corporation, San Jose, CA, 1998.

[17] *Field Programmable Gate Arrays*. San Jose, CA: Xilinx, Inc., 2002.

[18] Simulink, Natick, MA. [Online]. Available: http://www.mathworks.com.

[19] S. D. Haynes, J. Stone, P. Y. K. Cheung, and W. Luk, "Video image processing with the sonic architecture," *IEEE Computer*, vol. 33, pp. 50–57, Apr. 2000.

[20] B. L. Evans. Raster image processing on the TMS320C7X VLIW DSP. [Online]. Available: http://www.ece.utexas.edu/~bevans/hp-dsp-seminar/07_C6xImage2/sld001.htm.

[21] L. Hafer. BonsaiG. [Online]. Available: http://www.cs.sfu.ca/~lou/BonsaiG

[22] C. Lee, D. Kirovski, I. Hong, and M. Potkonjak, "DSP QUANT: Design, validation, and applications of DSP hard real-time benchmark," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, vol. 1, 1997, pp. 671–682.

[23] K. Parhi, *VLSI Digital Signal Process. Systems*. New York: Wiley, 1999.

[24] K.-I. Kum, J. Kang, and W. Sung, "AUTOSCALER for C: An optimizing floating-point to integer C program convertor for fixed-point digital signal processors," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 840–848, Sept. 2000.

[25] J. Frigo, M. Gokhale, and D. Lavenier, "Evaluation of the streams-C C-to-FPGA compiler: An applications perspective," in *Proc. ACM/SIGDA Int. Symp. FPGAs*, 2001, pp. 134–140.

[26] L. B. Jackson, "On the interaction of roundoff noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, vol. 49, pp. 159–184, 1970.

[27] G. A. Constantinides, "Perturbation analysis for word-length optimization," in *Proc. IEEE Symp. Field-Programmable Custom Comput. Mach.*, Napa, CA, Apr. 2003, pp. 81–90.