Optimal Combined Word-length Allocation and Architectural Synthesis of Digital Signal Processing Circuits

Gabriel Caffarena, *Student Member IEEE*, George A. Constantinides, *Member IEEE*,
Peter Y.K. Cheung, *Senior Member IEEE*, Carlos Carreras and Octavio Nieto-Taladriz

***Abstract*— In this paper we address the combined application of word-length allocation and architectural synthesis of digital signal processing systems for field programmable gate array devices. These two design tasks are traditionally performed sequentially, thus lessening the overall design complexity, but ignoring forward and backward dependencies that may lead to cost reductions. Mixed integer linear programming is used to formulate the combined problem and results are compared to the two-step traditional approach.**

***Index Terms*— Digital signal processing, word-length allocation, architectural synthesis, fixed-point arithmetic**

I. Introduction

THIS paper addresses the problem of hardware synthesis of digital signal processing (DSP) algorithms under both error and latency constraints. Programmable logic devices are chosen as the target architecture.

The multiple word-length implementation of DSP algorithms [1] has lately been an active research field. The traditional uniform word-length design approach, inherited from a microprocessor-oriented approach, has been reviewed for the last few years and algorithms for both word-length allocation [2-7] and architectural synthesis [2,5,8-10] have been tuned to the most efficient multiple word-length design. However, little research has been carried out regarding the combined application of both design tasks [5].

A mixed integer linear programming (MILP) formulation is presented aiming two objectives: *i)* to assess the suitability of the simultaneous application of these two design tasks; and *ii)* to present a reference landmark for the evaluation of future heuristic algorithms.

The main contribution of this paper is the presentation of an optimal analysis of the simultaneous application of word-length allocation and architectural synthesis. This approach is compared to the sequential application of optimal algorithms for word-length allocation and architectural synthesis. Area savings close to 20% are reported.

The paper is organized as follows: Section II deals with the main concepts involved in the combined application approach. The next section covers the MILP formulation of the problem. In section IV some results are analyzed. And finally, the conclusions are drawn in section V.

Manuscript received November 19, 2004. This work has been partially supported by the Spanish Ministry of Science and Technology under research project TIC2003-09061-C03-02 and by the Engineering and Physical Sciences Research Council, U.K.

G. Caffarena, C. Carreras and O. Nieto-Taladriz are with the Departamento de Ingenieria Electronica, Universidad Politecnica de Madrid, Madrid 28040, Spain. (e-mail: {g.caffarena, carreras, nieto}@die.upm.es).

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

II. Combined word-length allocation and architectural synthesis

A. Combined approach

The combined application of the word-length allocation and architectural synthesis tasks has as a starting point a computation graph $G(V,S)$, a maximum latency λ and a maximum noise variance at the output ε.

$G(V,S)$ is a formal representation of the algorithm, where V is a set of graph nodes representing operations and $S \subset V \times V$ is a set of directed edges representing signals that determine the data flow. We consider $V=V_G \cup V_A \cup V_D \cup V_F \cup V_I \cup V_O$ composed of gains, additions, unit delays, forks (branching nodes) and input and output nodes. Signals are in two's complement fixed-point format defined by the pair (n, p), where n is the word-length of the signal, excluding the sign bit; and p is the scaling of the signal, that represents the displacement of the binary point from the sign bit (see fig. 1).

Operations are to be implemented on resources from set R and it is the aim of the combined approach to find the word-lengths n, the time step when each operation is executed (*scheduling*), the types and number of resources forming R (*resource allocation*) and the binding between operations and resources (*resource binding*) that comply with both λ and ε constraints while leading to a minimum area implementation.

The notation adopted in this paper is as follows: $f(X)$ represents the range of a function $f: X \rightarrow Y$, $|X|$, the cardinality of set X, $\wedge$ denotes logical AND, $\vee$, logical OR, and \, set subtraction. The set of input signals driving node v are expressed as $prec(v)$ and the output signals driven by v as $succ(v)$. The upper bound on variable n are represented as $\hat{n}$.

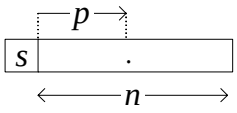

Fig. 1. Fixed-point format model: *n* is the signal word-length and *p* indicates the position of the fractionary point.

B. Scaling

The scaling of signals can be computed before the optimization process starts. We choose an analytical approach based on the computation of the l_1-norm for each signal. Given the input peak value |x| the scaling of a signal *s* is determined by (1), where $L_1(\cdot)$ is the l_1-norm and $\bar{H}_s(z)$ is the transfer function from the input to signal *s*.

$$p_s = \left\lfloor \log_2\left(|x| \cdot L_1\left\{\bar{H}_s(z)\right\}\right)\right\rfloor + 1 \tag{1}$$

C. Noise model

We adopt the quantization error model presented in [11]. The effect of the quantization error introduced by the quantization of a signal *s* from n_s^q bits to n_s bits ($n_s \le n_s^q$) can be modeled by the injection of a uniformly distributed white noise with a variance equal to (2). The variance of the noise contribution at the output due to this quantization is $L_2^2(H_s(z)) \cdot \sigma_s^2$, where $L_2(\cdot)$ is the l_2-norm and $H_s(Z)$ is the transfer function from signal *s* to the output.

$$\sigma_s^2 = 2^{-2p_s}\left(2^{-2n_s} - 2^{-2n_s^q}\right)/12 \tag{2}$$

As stated in [6], the error introduced by the outputs of forks requires a special treatment. A cascade model is used to take into account the correlation between outputs and the input of the fork (see fig. 2). The output signals of a w-way fork are arranged following a descendant order of word-lengths expressed by the w-tuple σ, where $n_\sigma(r) \ge n_\sigma(r+1)$. The error that a fork introduces can be expressed as in (3).

$$\begin{aligned} &\bigwedge_{r=1}^{w-1} n_{\sigma(r)} \ge n_{\sigma(r+1)} \Rightarrow \\ &E_v = 2^{2p_i}\Bigg(\sum_{r=1}^{w-1} L_2^2\left(\sum_{h=r+1}^{w} H_{\sigma(h)}\right)\left(2^{-2n_{\sigma(r+1)}} - 2^{-2n_{\sigma(r)}}\right) \\ &\quad + L_2^2\left(\sum_{h=1}^{w} H_{\sigma(h)}\right)\left(2^{-2n_{\sigma(1)}} - 2^{-2n_i^q}\right)\Bigg) \end{aligned} \tag{3}$$

D. Architectural synthesis

The data-flow of a single iteration of the algorithm is expressed by means of the sequencing graph $P(V,D)$ extracted from *G*. *V* is the set of operations and $D \subset V \times V$ are the edges specifying the precedence relations among operations. This graph is used to decide about scheduling.

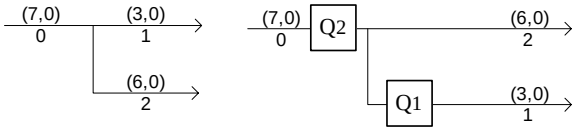

Fig. 2. Fork model: a) 2-way fork: b) cascade model

In our approach we assume 1-cycle latency operations. Each operation $v \in V$ can be executed during the time interval defined by $T(v)$ (4) where Z_+ denotes the set of non-negative integers. $ASAP(v)$ is the execution time of operation *v* for the *as soon as possible* scheduling and $ALAP(v,\lambda)$ is the execution time of operation *v* for the *as late as possible* scheduling for a total time steps of λ. The set of all possible execution times is given by (5).

$$T(v) = \{t \mid t \in Z_+ : t \ge ASAP(v) \wedge t \le ALAP(v,\lambda)\} \tag{4}$$

$$T = \{t \mid \exists v \in D : t \in T(v)\} = \{1..\lambda\} \tag{5}$$

The set of resources $R = R_M \cup R_A \cup R_R$ is divided into multipliers, adders and registers which implement gains, additions and delays. Multiplexing logic and memory to store intermediate values are not considered among resources.

We express the compatibility between an operation or set of operations and resources with function $R(V)$: $V \rightarrow R$.

Targeting programmable logic devices, we only regard multipliers as shareable resources, since the multiplexing logic necessary is often negligible compared to the area of these resources, a situation that does not apply to adders or registers. Thus, there are dedicated resources to implement each addition and delay so $R_A(V_A)$ and $R_R(V_D)$ are one-to-one functions, and $|R_A| = |V_A|$ and $|R_R| = |V_D|$.

Multipliers have one input devoted to coefficients and its word-length is equal to a system-wide constant, *cw*. The other input is assigned to the input signal of gains and must have a word-length greater than or equal to the maximum word-length of the input signals of gains bound to the resource.

Since we aim an optimal solution of the problem it is possible to estimate the exact number of multipliers required. The tight lower bound on resources presented in [12] can be used to obtain this estimation. This bound is computed as follows: *i)* enumerate all possible time intervals $T_i \subseteq T$; *ii)* for each *Ti* generate the set $V_{G,i}' \subseteq V_G$ composed of all gains whose execution intervals are included within T_i; *iii)* calculate a lower bound for each T_i using formula $\lceil |V_{G,i}|/|T_i| \rceil$; and *iv)* set the lower bound as the maximum of all bounds computed.

Initially, all gains can be implemented on all multipliers, therefore $R_M(V_G) = R_M$.

E. Area models

The cost of adder $r \in R_A$ bound to addition $v \in V_A$ with inputs *i* and *j* and output *o* is given by (6) and it is taken from [6]. A

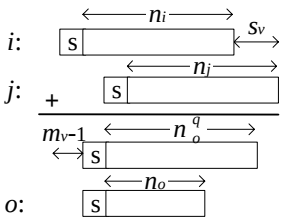

Fig. 3. Example of configuration of adder signals.

ripple-carry adder is supposed and the two possible cases in (6) correspond to a sum-and-carry full adder and to a carry-only full adder. Signals *i* and *j* must comply with the following: signal *j* is shifted s_v bits from the least significant bit of *i* and scaling p_i of signal *i* should be bigger than or equal to the value of p_j (see fig. 3 for an example). In (6), m_v is the number of non-required bits at the output due to scaling (7). Note that the *max* operation in (8) can be expressed as a disjunction. Constants k_1 and k_2 are technology dependent.

$$\forall r \in R_A, \forall v \in V_A : R_A(v) = r,$$
$$\text{cost}(r) = \begin{cases} k_1(n_o + 1) + k_2(\beta_v - m_v - n_o + 1), \\ \qquad \text{if } n_o + m_v \le \beta_v + 1 \\ k_1[\beta_v - m_v + 2], \text{otherwise} \end{cases} \quad (6)$$

$$m_v = \max(p_i, p_j) + 1 - p_o \quad (7)$$

$$\beta_v = \max(n_j - s_v, n_i) = \begin{cases} n_j + p_j - p_i, \text{if } ni - nj + pj - pi \ge 0 \\ n_i, \text{otherwise} \end{cases} \quad (8)$$

The cost of a register $r \in R_R$ bound to delay $v \in V_D$ with input *i* is given by (9). Again, k_3 is a technology dependent constant.

$$\forall r \in R_R, \forall v \in V_D : R_R(v) = r, \ \text{cost}(r) = k_3 \cdot n_i \quad (9)$$

Eqn. (10) contains the cost of a multiplier $r \in R_M$ bound to a subset of gains $V_G' \subseteq V_G$ with inputs $S' = prec(V_G') \subseteq S$.

$$\text{cost}(r) = (\max(n(S')) + 1) \cdot (cw + 1) \quad (10)$$

III. MILP FORMULATION

This section relies on some knowledge of integer linear programming [13]. The variables used in the MILP model are divided into: binary scheduling and resource binding variable (*x*), integer signal word-lengths (*n*), integer signal word-lengths before quantization (n^q), binary auxiliary signal word-length ($\bar{n}$), binary auxiliary signal word-lengths before quantization ($\bar{n}^q$), binary decision variables (δ, ε and η), real adder costs (*A*), integer auxiliary variables (β) and real fork node error variables (*E*).

In the following subsections we present the formulation of the MILP model.

A. Objective function

The objective function is the sum of the area of all resources (adders, registers and multipliers) and it is given by (11). The cost of adders A_r is to be linearized in the constraints section according to (6).

$$\min: \sum_{r \in R_A} A_r + \sum_{r \in R_R} k_3 \cdot n_{in(r)} + \sum_{r \in R_M} (n_r + 1)(cw + 1) \quad (11)$$

B. Architectural synthesis constraints

Here we introduce the constraints related to scheduling, resource allocation and resource binding. Eqn. (12) defines the binary variables $x_{v,t,r}$ that steer the constraints in this subsection.

$$x_{v,t,r} = \begin{cases} 1, & \text{if operation } v \text{ is scheduled at time-step } t \\ & \text{on resource } r \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

Eqn. (13) shows the binding constraint that ensures that an operation is executed on exactly one resource. The next constraint (14) states that a resource does not implement more than one operation at a time. Note that there is no need to apply (13) and (14) to operations with dedicated resources. The precedence constraints are given by (15) ensuring that operations obey the dependencies in the sequencing graph *P*.

$$\forall v \in V_G, \sum_{r \in R(v)} \sum_{t \in T(v)} x_{v,t,r} = 1 \quad (13)$$

$$\forall t \in T, \forall r \in R_M, \sum_{v \in V : r \in R(v)} \sum_{t_1 \in \{t\} \cap T(v)} x_{v,t_1,r} \le 1 \quad (14)$$

$$\forall (v_1, v_2) \in D, \forall t \in T(v_2) \cap T(v_1)$$
$$\sum_{r \in R(v_2)} \sum_{t_2 \in T(v_2) : t_2 \le t} x_{v_2,t_2,r} + \sum_{r \in R(v_1)} \sum_{t_1 \in T(v_1) : t_1 \ge t} x_{v_1,t_1,r} \le 1 \quad (15)$$

And finally, (16) expresses the resource compatibility constraints, which guarantee that a resource bound to several operations must be compatible with all of them. Again, only multipliers are considered: the input devoted to signals must have a word-length as big as the maximum of the word-lengths of each gain input bound to it. The summation $\sum x_{v,t,r}$ expresses the resource binding and it is equal to 1 if operation *v* is bound to resource *r*.

$$\forall r \in R_M, \forall v \in V_G, \ n_{prec(v)} - n_r \le \hat{n}_{prec(v)} \left(1 - \sum_{t \in T(v)} x_{v,t,r}\right) \quad (16)$$

Note that although only multipliers are prone to sharing, the notation can be easily extended to include more resources that can be shared (dividers, adders, etc), or to map more than one type of operation to the same resource (e.g. gains and multiplications bound to multipliers).

C. Adder cost

The linearization of the adder cost is taken from [6] and presented in (17) to (24) for completeness. Constraints (17)-(20) cast (8) using binary decision variables δ_{a1} and δ_{a2}, and also trivial bounds on the left side of the equations. Similarly, the disjunction in (7) is cast by (21)-(24), where binary variables δ_{a3} and δ_{a4} and trivial bounds are introduced.

$$n_{i_1} - n_j + p_j - p_i < \delta_{a1}\left(\hat{n}_{i_1} + p_j - p_i\right) \tag{17}$$

$$\beta_v - n_j + p_j - p_i < (1-\delta_{a1})\left(-\hat{n}_j + p_j - p_i\right) \tag{18}$$

$$n_i - n_j + p_j - p_i \geq \delta_{a2}\left(-\hat{n}_j + p_j - p_i\right) \tag{19}$$

$$\beta_v - n_j \geq (1-\delta_{a2})\left(-\hat{n}_j\right) \tag{20}$$

$$n_o - \beta_v + m_v - 1 \geq \delta_{a3}\left(m_v - \hat{\beta}_v\right) \tag{21}$$

$$\begin{aligned} &A_r + (k_2 - k_1)n_o - k_2\beta_v + k_2(m_v - 1) - k_1 \geq \\ &(1-\delta_{a3})\left[(k_2 - k_1)\hat{n}_o - k_2\hat{\beta}_v + k_2(m_v - 1) - k_1\right] \end{aligned} \tag{22}$$

$$n_o - \beta_v + m_v - 1 < \delta_{a4}\left(\hat{n}_o + m_v - 2\right) \tag{23}$$

$$A_r + k_1\left(m_v - \beta_v - 2\right) \geq (1-\delta_{a4})k_1\left(m_v - \hat{\beta}_v - 2\right) \tag{24}$$

D. Word-length allocation constraints

Here we present the constraints related to the estimation of the noise at the output of the system [6]. The error constraint is given by (25) and it is divided into two summations, the first dealing with forks signals and the second dealing with the remaining signals in *S*.

$$\sum_{v \in V_F} E_v + \sum_{s \in S \setminus prec(V_F) \setminus succ(V_F)} 2^{2p_s} L_2^2\left(H_s(Z)\right)\left(2^{-2n_s} - 2^{-2n_s^q}\right) \leq 12\varepsilon \tag{25}$$

Note that non-constant powers of two must be linearized. Each 2^{-2n} term is replaced by (26), where $\bar{n}_b$ are binary auxiliary variables associated to signal *n* by (27) and (28). For simplification' sake we leave all non-constant powers of two unchanged in this text.

$$\sum_{b=1}^{\hat{n}} 2^{-2b}\bar{n}_b \tag{26}$$

$$n - \sum_{b=1}^{\hat{n}} b \cdot \bar{n}_b = 0 \tag{27}$$

$$\sum_{b=1}^{\hat{n}} \bar{n}_b = 1 \tag{28}$$

The noise E_v introduced by a fork *v* is expressed by constraints (29)-(32), which come from applying DeMorgan's theorem to (3) and linearizing the disjunction obtained. Binary variables η and ε are introduced. These constraints are repeated for each possible ordering σ of the outputs of a fork.

$$n_{\sigma(1)} - n_{\sigma(2)} < \varepsilon_{v\sigma(1),\sigma(2)}\hat{n}_{\sigma(1)} \tag{29}$$

...

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

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

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

E. Conditioning constraints

This last set of constraints computes the word-lengths before quantization when considering scaling and word-length propagation information [6].

The prequantization word-length of the output of an addition *v* with inputs *i* and *j* and output *o* is equal to $max(n_i - p_i + p_o, n_j - p_j + p_o)$, expression linearized through (33) and (34).

$$n_o^q \geq n_i - p_i + p_o \tag{33}$$

$$n_o^q \geq n_j - p_j + p_o \tag{34}$$

Delays *v* with input *i* are conditioned through (35).

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

Regarding forks, the outputs do not require conditioning but its inputs must comply with (36).

$$\forall_{s \in succ(v)} n_i \geq n_s \tag{36}$$

The conditioning of gain *v* is expressed by constraint (37), where $p_{cw}(v)$ is the scaling of the coefficient associated to *v*.

$$n_o^q = n_i + cw - p_i - p_{cw}(v) + p_o \tag{37}$$

Finally, (38) indicates that signals must be truncated to a word-length smaller than or equal to its prequantization word-length.

$$\forall s \in S : \exists (n_s, n_s^q), n_s \leq n_s^q \tag{38}$$

F. Bounds on word-length of variables

Bounds on word-lengths are estimated using an adaptation of the procedure presented in [6]: *i)* use an heuristic algorithm to allocate word-lengths and calculate the area *A* due to gains; *ii)* assign to each gain input the word-length that makes its area

be as big as *A*; *iii)* set all gain inputs to the maximum word-length of all gain inputs; and *iv)* condition the graph.

IV. RESULTS

An MILP solver [14] was used to find the optimal solutions for a set of FIR filters with 6-bit inputs and 4-bit coefficients: a 2nd order low-pass filter (LP_2), a 4th order low-pass filter (LP_4) and a 2nd order high-pass filter (HP_4). Two latencies were used: a latency L_{MIN} which was the minimum latency that allowed resource sharing, and a latency L_{MAX} which was the minimum latency that allowed maximum resource sharing. For each latency conditions two solutions were computed, one for the *sequential approach*, where the error constrained problem was solved first and its solution was fed to the latency constrained problem, and another for the *combined approach*. The comparison results are in table 1 in terms of percentage of area reduction comparing the combined approach to the sequential approach. The area savings range from 0% to 17.7%. The area savings are due to an optimal exploration of the dependencies between word-lengths, resources and error variance. For instance, the LP_2 filter requires only one multiplier, and the sequential approach assigns world-lengths of 1 and 4 bits to the gain inputs bound to it for an error bound of 1e-1, while the combined approach assigns a uniform word-length of 2 bits. Thus, the area of the multiplier is significantly reduced and the additional noise is compensated by increasing slightly the word-lengths of signals related to adders and registers. The total area is reduced. For an error bound of 1e-2 the sequential solution assigns world-lengths of 1 and 4 bits to the gains bound to the multiplier, while the combined solution assigns a uniform word-length of 4 bits. Thus, the area of the multiplier is not reduced but the noise is decreased allowing to reduce the precision of signals related to adders and registers with the consequent area saving.

Fig. 4 depicts the trade-off curves of the area in logic cells versus the error variance for cases LP_2 and HP_4. The graphs yield that the combined approach always produces results equal to or better than those of the sequential approach.

V. CONCLUSIONS

In this paper we have presented a novel MILP formulation for the combined error and latency constrainted area-minimization problem applicable to LTI DSP algorithms. This optimal model of the problem can be used to assess the quality of future heuristic methods addressing such problem. The problem can be easily reformulated to include some types of non-linear systems [15].

Results show the advantage produced by the combined use of these well-known design tasks. Area savings up to 17% are reported.

TABLE I
AREA REDUCTION (%) OBTAINED BY THE COMBINED APPROACH

Bound on Error Variance	LP_2 L_{MIN}=L_{MAX}	LP_4 L_{MIN}	LP_4 L_{MAX}	HP_4 L_{MIN}	HP_4 L_{MAX}
3e-3	0.0	5.4	4.0	6.6	3.7
5e-3	5.8	3.7	0.0	0.7	0.8
1e-2	6.4	0.0	0.0	7.1	6.9
3e-2	13.5	0.0	1.3	0.0	0.0
5e-2	0.0	0.0	0.0	4.7	4.5
1e-1	17.7	7.5	9.4	0.00	0.0

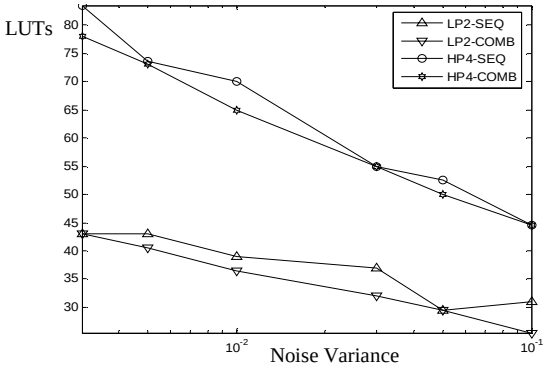

Fig. 4. Area/error tradeoff curves for a 2nd order FIR low-pass (LP2) and 4th order FIR high-pass (HP4) under minimum sharing latency, for both sequential (SEQ) and combined (COMB) approaches.

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, 2001.

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

[3] 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.

[4] C. Carreras, J.A. Lopez and O. Nieto-Taladriz, "Bit-width selection for data-path implementations," in Proc. Int. Symp. System Synthesis, San Jose, CA, 1999, pp. 114 – 119.

[5] K.-I. Kum and W. Sung, "Combined word-length optimization and highlevel synthesis of digital signal processing systems," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 921–930, Aug. 2001.

[6] G.A. Constantinides, P.Y.K. Luk and W. Luk, "Wordlength Optimization for Linear Digital Signal Processing, " *IEEE Trans. Computer-Aided Design*, vol. 22, pp. 1432-1442, Oct. 2003.

[7] G. Caffarena, A. Fernandez, C. Carreras and O. Nieto-Taladriz, "Fixed-point refinement of OFDM-based adaptive equalizers: an heuristic approach," in *Proc. Eur. Signal Processing Conf.*, Vienna, Austria, 2004, pp. 1353–1356.

[8] J.-I. Choi, H.-S. Jun and S.-Y. Hwang, "Efficient hardware optimization algorithm for fixed point digital signal processing ASIC design," *IEE Electron. Lett.*, vol.32, pp. 992-994, 1996.

[9] G.A. Constantinides, P.Y.K. Luk and W. Luk, "Heuristic Datapath Allocation for Multiple Wordlength Systems," in *Proc. Design Automation Test Eur*., Munich, Germany, 2001, pp. 791–796.

[10] S. Mahlke, R. Ravindran, M. Schlansker, R. Schreiber and T. Sherwood, "Bitwidth cognizant architecture synthesis of custom hardware accelerators," *IEEE Trans. Computer-Aided Design*, vol. 20 , pp. 1355-1371, Nov. 2001.

[11] G.A. Constantinides, P.Y.K. Cheung and W. Luk, "Truncation noise in fixed-point SFG's," *IEE Electron. Lett*., vol. 35, no. 23, pp. 2012–2014, 1999.

[12] Seong Yong Ohm, F.J. Kurdahi and N.D. Dutt, "A unified lower bound estimation technique for high-level synthesis," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 458-472, May 1997.

[13] R.S. Garfinkel and G.L. Nemhauser, *Integer Programming*, John Wiley and sons, New York, 1972.

[14] Mosek Aps, http://www.mosek.com

[15] G.A. Constantinides, "Perturbation Analysis for Word-length Optimization," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, 2003, pp. 81-90.