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 brief, we address the combined application of word-length allocation and architectural synthesis of linear time-invariant digital signal processing systems. 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*—Architectural synthesis, digital signal processing, fixed-point arithmetic, word-length allocation.**

I. Introduction

THIS brief 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]–[6] and architectural synthesis [4], [7], [8] have been tuned to the more efficient multiple word-length design. However, little research has been carried out regarding the combined application of both design tasks. In [4] a 3-step methodology is presented: approximate word-length allocation, architectural synthesis and accurate word-length allocation of the resulting architecture. The approach is a pioneer work in the combination of word-length allocation and architectural synthesis; it only lacks a report on the area savings obtained compared to the traditional approach.

Mixed integer linear programming (MILP) is used to define the problem. Mainly, it allows assessing the suitability of the simultaneous application of these two design tasks, and the results presented in this brief can be used to evaluate future heuristic algorithms.

Manuscript received January 28, 2005; revised July 29, 2005. This work was supported in part 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. This paper was recommended by Associate Editor C.-T. Lin.

G. Caffarena, C. Carreras, and O. Nieto-Taladriz are with the Departamento de Ingeniería Electrónica, Universidad Politécnica de Madrid, Madrid 28040, Spain (e-mail: gabriel@die.upm.es; carreras@die.upm.es; nieto@die.upm.es).

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

Digital Object Identifier 10.1109/TCSII.2005.862175

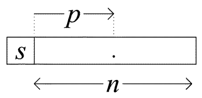

Fig. 1. Fixed-point format model: n is the signal word-length and p indicates the position of the fractionary point with respect to the sign bit s.

The main contribution of this brief 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 up to a 13% are reported.

The brief is divided as follows. Section II deals with the main concepts involved in the combined application approach. The next section deals with the MILP formulation of the problem. In Section IV some results are analyzed. Finally, conclusions are drawn in Section V.

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 determines 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 not including 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 achieving minimum area.

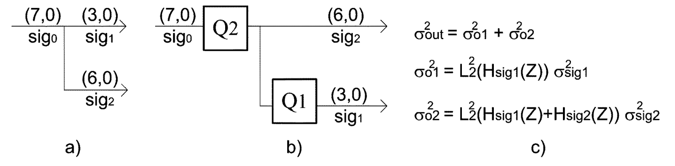

Fig. 2. Fork model. (a) 2-way fork. (b) Cascade model with sorted outputs. (c) Noise at the output (expressed as σ^2_{out}) due to quantizations Q1 and Q2.

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

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

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

C. Noise Model

We adopt the quantization error presented in [9]. The quantization error introduced by the quantization of a signal s from n_s^q bits to n_s bits ($n_s \leq n_s^q$) is modeled by the injection of a uniform-distributed white noise with a variance equal to (2). The variance of the noise contribution at the output 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

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

As stated in [5], the error introduced by forks requires a special treatment. Fig. 2 shows a 2-way fork with quantized outputs. First, the outputs must be sorted in descendant word-length order to take into account the correlation between them and the input of the fork [Fig. 2(b)]. It can be clearly seen that the quantization noise injected by Q_2 traverses both sig_1 and sig_2, hence σ^2_{o2} requires both H_{sig1} and H_{sig2} [Fig. 2(c)]. The noise injected by Q_1 only traverses sig_1, thus only H_{sig1} contributes to the output's noise. The error that a w-way fork introduces can be expressed as in (3), where the w-tuple α expresses the order of the outputs [5]

$$\begin{aligned} &\wedge_{r=1}^{w-1} n_{\alpha(r)} \geq n_{\alpha(r+1)} \Rightarrow \\ &E_v = \frac{2^{2p_i}}{12} \left(\sum_{r=1}^{w-1} L_2^2 \left(\sum_{h=r+1}^{w} H_{\alpha(h)} \right) \left(2^{-2n_{\alpha(r+1)}} - 2^{-2n_{\alpha(r)}}\right) \right. \\ &\qquad \left. + L_2^2 \left(\sum_{h=1}^{w} H_{\alpha(h)} \right) \left(2^{-2n_{\alpha(1)}} - 2^{-2n_i^q}\right) \right). \end{aligned} \tag{3}$$

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.

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 nonnegative integers. $\text{ASAP}(v)$ is the execution time of operation v for the *as soon as possible* scheduling and $\text{ALAP}(v, \lambda)$ 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

$$T(v) = \{t | t \in Z_+ : t \geq \text{ASAP}(v) \wedge t \leq \text{ALAP}(v, \lambda)\} \tag{4}$$

$$T = \{t | \exists v \in V : t \in T(v)\} = \{1 \ldots \lambda\}. \tag{5}$$

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 regard as shareable only multipliers 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. For instance, the ratio between the area of a LUT-based 16×16-bit multiplier and a 16-bit 2-input multiplexer and a 16-bit 4-input multiplexer are 17.25 and 8.625 respectively for Virtex-2 and Virtex-4 devices (using Xilinx ISE v7.1). 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 n_{coeff}. 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 inputs of gains bound to the resource.

An upper bound on the number of multipliers necessary can be estimated from the number of multipliers necessary to implement the ASAP scheduling. Initially, all gains can be implemented on all multipliers, therefore $R_M(V_G) = R_M$.

E. Area Models

The cost of an 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 derived from the model in [5]. A ripple-carry adder is supposed. 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 should be bigger than or equal to the value of p_j (see Fig. 3 for an example). Equation (6) requires the definition of β and m_v. Let us define β as the number of overlapped bits between i and j with sign extension (7). A *safe* adder would require $\beta + 1$ bits. Let us define m_v as the number of nonrequired bits at the output due to scaling (8). The area of an *optimized* adder is equal to the area of the *safe* adder minus m_v bits (6). Note that the *max* operation in (7) can be expressed as a disjunction, and that m_v is a constant number

$$\begin{aligned} &\forall r \in R_A \qquad \forall v \in V_A : R_A(v) = r \\ &\text{cost}(r) \quad = (\beta_v + 1) - m_v \end{aligned} \tag{6}$$

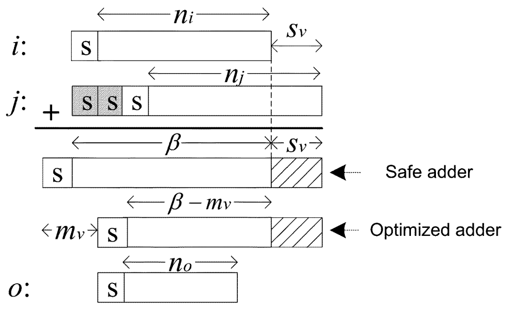

Fig. 3. Example of configuration of addition signals.

$$\begin{aligned}\beta_v &= \max(n_j - s_v, n_i) + 1 \\ &= \begin{cases} n_j + p_j - p_i + 1, & \text{if } n_i - n_j + p_j - p_i \geq 0 \\ n_i + 1, & \text{otherwise}\end{cases}\end{aligned} \tag{7}$$

$$m_v = (\max(p_i, p_j) + 1) - p_o. \tag{8}$$

The cost of a register $r \in R_R$ bound to delay $v \in V_D$ with input i is given by the straightforward equation

$$\forall r \in R_R \qquad \forall v \in V_D : R_R(v) = r, \text{cost}(r) = n_i. \tag{9}$$

Equation (10) contains the cost of a multiplier $r \in R_M$ bound to a subset of gains $V'_G \subseteq V_G$ with inputs $S' = \text{prec}(V'_G) \subseteq S$

$$\text{cost}(r) = (\max(n(S')) + 1) \cdot (n_{\text{coeff}} + 1). \tag{10}$$

III. MILP Formulation

This section relies on some knowledge of integer linear programming [10]. The variables used in the MILP model are divided into: binary scheduling and resource binding variables (x), integer signal word-lengths (n), integer signal word-lengths before quantization (n^q), binary auxiliary signal word-lengths ($\bar{n}$), binary auxiliary signal word-lengths before quantization ($\bar{n}^q$), binary decision variables (δ, ε and η), integer 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)

$$\min: \sum_{r \in R_A} A_r + \sum_{r \in R_R} n_{\text{in}(r)} + \sum_{r \in R_M} (n_r + 1)(n_{\text{coeff}} + 1). \tag{11}$$

B. Architectural Synthesis Constraints

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

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

Equation (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

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

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

$$\begin{aligned}&\forall (v_1, v_2) \in D \qquad \forall t \in T(v_2) \cap T(v_1) \\ &\sum_{r \in R(v_2)} \sum_{t_2 \in T(v_2): t_2 \leq t} x_{v_2,t_2,r} \\ &\quad + \sum_{r \in R(v_1)} \sum_{t_1 \in T(v_1): t_1 \geq t} x_{v_1,t_1,r} \leq 1.\end{aligned} \tag{15}$$

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 $\sum x_{v,t,r}$ is equal to 1 if operation v is bound to resource r

$$\begin{gathered}\forall r \in R_M \qquad \forall v \in V_G \\ n_{\text{prec}(v)} - n_r \leq \hat{n}_{\text{prec}(v)} \left(1 - \sum_{t \in T(v)} x_{v,t,r}\right).\end{gathered} \tag{16}$$

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 based on the model from [5]. Constraints (17)–(20) cast (7) using binary decision variables δ_{a1} and δ_{a2}, and also trivial bounds on the left side of the equations. Equation (6) is directly implemented using constraint (21)

$$n_i - n_j + p_j - p_i < \delta_{a1}(\hat{n}_i + p_j - p_i) \tag{17}$$

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

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

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

$$A_r = \beta_v + 1 - m_v. \tag{21}$$

D. Word-Length Allocation Constraints

Here, we present the constraints related to the estimation of the noise at the output of the system [5]. The error constraint is given by (22) and it is divided into two summations, the first dealing with forks' signals, and the second dealing with the remaining signals in S (see Section II-C)

$$\sum_{v\in V_F} E_v + \sum_{s\in S\setminus \mathrm{prec}(V_F)\setminus \mathrm{succ}(V_F)} \frac{2^{2p_s}}{12} L_2^2(H_s(Z)) \times \left(2^{-2n_s} - 2^{-2n_s^q}\right) \le \varepsilon. \quad (22)$$

Note that nonconstant powers of two must be linearized. Each 2^{-2n} term is replaced by $\sum_{b=1}^{\hat{n}} 2^{-2b}\bar{n}_b$, where $\bar{n}_b$ are binary auxiliary variables associated to signal n by (23) and (24). For simplification sake, we leave all nonconstant powers of two as they are throughout the text

$$n - \sum_{b=1}^{\hat{n}} b\cdot \bar{n}_b = 0 \quad (23)$$

$$\sum_{b=1}^{\hat{n}} \bar{n}_b = 1. \quad (24)$$

The noise E_v introduced by a fork v is expressed by constraints (25)–(28), 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 α of the outputs of a fork

$$n_{\alpha(1)} - n_{\alpha(2)} < \varepsilon_{v\alpha(1),\alpha(2)}\hat{n}_{\alpha(1)}\cdots \quad (25)$$

$$n_{\alpha(w-1)} - n_{\alpha(w)} < \varepsilon_{v\alpha(w-1),\alpha(w)}\hat{n}_{\alpha(w-1)} \quad (26)$$

$$E_v - \frac{2^{2p_i}}{12}\sum_{r=1}^{w-1} L_2^2\left(\sum_{h=r+1}^{w} H_{\alpha(h)}\right)\left(2^{-2n_{\alpha(r+1)}} - 2^{-2n_{\alpha(r)}}\right) - \frac{2^{2p_i}}{12} L_2^2\left(\sum_{h=1}^{w} H_{\alpha(h)}\right)\left(2^{-2n_{\alpha(1)}} - 2^{-2n_i^q}\right) \ge -\eta_{v\alpha}\frac{2^{2(p_i-1)}}{12}\sum_{r=0}^{w-1} L_2^2\left(\sum_{h=r+1}^{w} H_{\alpha(h)}\right) \quad (27)$$

$$\sum_{r=1}^{w-1} \varepsilon_{v\alpha(r),\alpha(r+1)} + \eta_{v\alpha} \le w-1. \quad (28)$$

E. Conditioning Constraints

This last set of constraints computes the word-lengths before quantization when considering scaling and word-length propagation information [5].

Given an addition v with inputs i and j and output o, its output's word-length is equal to $\max(n_i - p_i + p_o, n_j - p_j + p_o)$, expression linearized through the following:

$$n_o^q \ge n_i - p_i + p_o \quad (29)$$

$$n_o^q \ge n_j - p_j + p_o. \quad (30)$$

Delays v with input i are conditioned through

$$n_o^q = n_i. \quad (31)$$

Regarding forks, the outputs do not require conditioning but its inputs must comply with the following:

$$\forall s \in \mathrm{succ}(v), \qquad n_i \ge n_s. \quad (32)$$

The conditioning of gain v is expressed by constraint

$$n_o^q = n_i + n_{\mathrm{coeff}} - p_i - p_{\mathrm{coeff}}(v) + p_o \quad (33)$$

where $p_{\mathrm{coeff}}(v)$ is the scaling of the coefficient associated to v.

Finally,the following equation:

$$\forall s \in S : \exists(n_s, n_s^q), n_s \le n_s^q \quad (34)$$

indicates that signals must be truncated to a word-length smaller than or equal to its pre-quantization word-length.

F. Bounds on Word-Length of Variables

Bounds on word-lengths are estimated using an adaptation of the procedure presented in [5]: 1) use an heuristic algorithm to allocate word-lengths and calculate the area A due to gains; 2) assign to each gain input the word-length that makes its area to be as big as A; 3) set all gain inputs to the maximum word-length of all gain inputs; and 4) condition the graph.

IV. Results

An MILP solver [11] was used to find the optimal solutions for a set of FIR and IIR filters. The filters coefficients were obtained using the tool *fdatool* from Matlab 6.5 [12]. The FIR filters were implemented using the direct transposed symmetric FIR structure. We denote FIR_2 a second-order FIR filter with 8-bit inputs and 4-bit coefficients $\mathrm{B_{FIR2}} = [0.1759, 0.8, 0.1759]$; FIR_3 a third-order FIR filter with 8-bit inputs and 8-bit coefficients $\mathrm{B_{FIR3}} = [-0.1172, 0.6013, 0.6013, -0.1172]$; and FIR_4 a fourth-order FIR filter with 4-bit inputs and 4-bit coefficients $\mathrm{B_{FIR4}} = [-0.1210, -0.1423, 0.85, -0.1423, -0.1210]$. The IIR filter was a second-order filter with 4-bit inputs, gain $\mathrm{G} = 0.307\,089$ and 4-bit coefficients $\mathrm{SOS} = [1.0, 1.9999, 0.9999, 1.0, 0.064\,095\,5, 0.314]$, implemented using the direct form II transposed. The filters were tested under different latencies and for each latency 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 I in terms of percentage of area reduction comparing both sequential and combined approaches. The number of lookup tables (LUTs) required for the different approaches is also provided (*sequential/combined*). The area savings range from 0% to 13.16%, and are due to an optimal exploration of the dependencies between word-lengths, resources and error variance. Empty cells imply that a solution was not found by the MILP solver in practical times (less than 12 hours). For instance, Table II, shows the word-lengths, including the sign bit, assigned to gains ($g1-3$ and $g2$) and to multipliers ($m1$), adders ($a1, a2$ and $a3$) and registers ($r1, r2$ and $r3$) for the error/latency conditions A, B and C (see Table I, third row)

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

Error	**FIR$_2$_8x4** L=4	**FIR$_3$ 8x8** L=4	**FIR$_4$ 4x4** L=4	**IIR 4x4** L=4	**IIR 4x4** L=6
1.0e-5	0.00 (62/62)	**6.72** (119/111)	0.00 (58/58)	▨	▨
1.1e-5	**4.84** (62/59)	**5.93** (118/111)	0.00 (58/58)	▨	▨
3.3e-5	0.00 (55/55)	**1.90** (105/103)	0.00 (57/57)	▨	▨
5.0e-5	0.00 (54/54)	**7.62** (105/97)	0.00 (57/57)	▨	▨
1.0e-4	0.00 (48/48)	**1.08** (92/91)	0.00 (56/56)	▨	▨
1.1e-4	0.00 (47/47)	**1.08**^{C} (92/91)	0.00 (56/56)	▨	▨
3.3e-4	0.00 (40/40)	**2.50** (80/78)	**5.56** (54/51)	▨	▨
5.0e-4	0.00 (39/39)	0.00 (77/77)	0.00 (50/50)	0.00 (80/80)	▨
1.0e-3	**10.53** (38/34)	**11.84** (76/67)	0.00 (48/48)	**7.69** (78/72)	**3.70** (54/52)
1.1e-3	**13.16** (38/33)	**5.80**^{B} (65/69)	**7.84** (51/47)	**9.09** (77/70)	**5.66** (53/50)
3.3e-3	0.00 (30/30)	**12.70** (63/55)	**4.65** (43/41)	0.00 (64/64)	0.00 (44/44)
5.0e-3	**10.71** (28/25)	0.00 (51/51)	0.00 (37/37)	0.00 (56/56)	0.00 (40/40)
1.0e-2	0.00 (23/23)	**12.00** (50/44)	**2.78** (36/35)	0.00 (52/52)	0.00 (36/36)
1.1e-2	0.00 (23/23)	**12.24**^{A} (49/43)	**2.78** (36/35)	0.00 (52/52)	0.00 (36/36)
3.3e-2	0.00 (17/17)	**5.40** (37/35)	0.00 (25/25)	**13.04** (46/40)	**6.67** (30/28)
5.0e-2	0.00 (16/16)	0.00 (29/29)	0.00 (24/24)	0.00 (38/38)	0.00 (26/26)
1.0e-1	0.00 (16/16)	0.00 (28/28)	0.00 (23/23)	**11.43** (35/31)	0.00 (23/23)

TABLE II
DETAILED WORD-LENGTH DISTRIBUTION FOR FIR$_3$

%	**g1-3**	**g2**	**m1**	**r1**	**r2**	**r3**	**a1**	**a2**	**a3**
12.24^{A}	2x8→3x8	4x8→3x8	4x8→3x8	3	3	3	5	4→5	5→6
5.80^{B}	3x8→5x8	5x8	5x8	5→4	5→4	5→4	7→6	5	8
1.08^{C}	5x8→7x8	7x8	7x8	6	6	6	8	7→6	9

for FIR_3. The first row represents the area saving and states the error/latency condition. The rest of rows show the word-lengths, showing two word-lengths if the sequential results differ from the combined results $(\text{sequential} \rightarrow \text{combined})$. In case A the area of the multiplier $m1$ is reduced while the area of adders is slightly increased. In case B the area of registers and adders is reduced thanks to the increase of the word-lengths of gains. Finally, case C shows that an increase in the word-length of gains enables reducing the area of adders.

The execution times to solve the MILP problems range from several seconds (FIR_2) to several hours (IIR).

V. CONCLUSION

In this brief we have presented a novel MILP formulation for the combined error and latency constrained area-minimization problem, applicable to linear time-invariant DSP algorithms. This optimal model of the problem can be used to assess the quality of future heuristic methods that address the problem. The problem can be easily reformulated to include more complex resource binding [8], [13] that support multiple latency and pipelined resources, operation chaining, etc. The approach can be also applied to ASIC implementations.

Results show the advantage produced by the combined use of these well-known design tasks. Area savings up to 13% are reported.

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, pp. 51–60.

[2] 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.

[3] 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.

[4] K.-I. Kum and W. Sung, "Combined word-length optimization and highlevel synthesis of digital signal processing systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits*, vol. 20, no. 8, pp. 921–930, Aug. 2001.

[5] G. A. Constantinides, P. Y. K. Luk, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. Comput.-Aided Design Integr. Circuits*, vol. 22, pp. 1432–1442, Oct. 2003.

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

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

[8] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Optimal datapath allocation for multiple-wordlength systems," *Electron. Lett.*, vol. 36, pp. 1508–1509, 2000.

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

[10] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*. New York: Wiley, 1972.

[11] Mosek Aps (2004). [Online]. Available: http://www.mosek.com

[12] *Using FDATool with the Filter Design Toolbox.*. Natick, MA: The Mathworks Inc., 2004.

[13] B. Landwehr, P. Marwedel, and R. Dömer, "OSCAR: Optimum simultaneous scheduling, allocation and resource binding based on integer programming," in *Proc. Design Automation Test Eur.*, Grenoble, France, 1994, pp. 90–95.